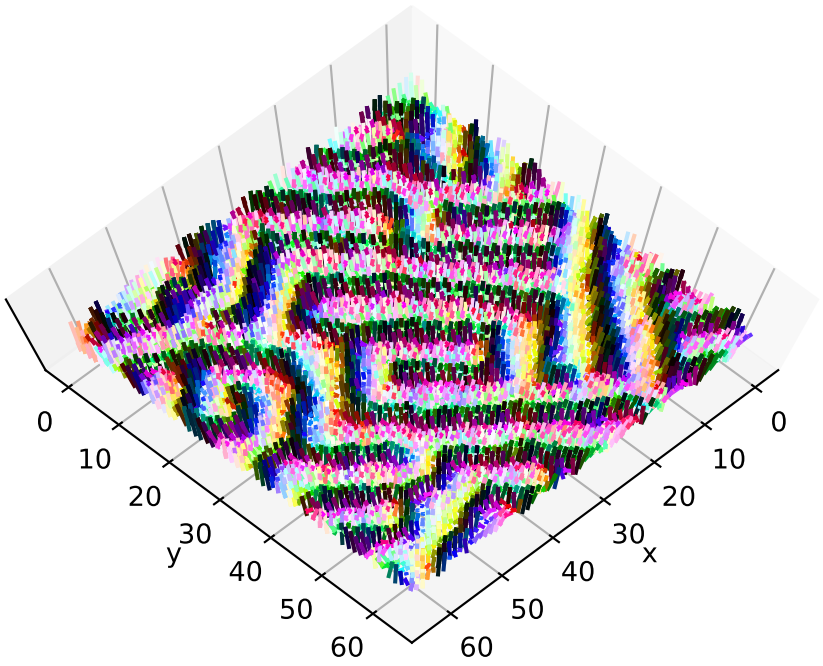
A Monte Carlo simulation of classical microscopic models for magnetic materials

Andrea Maiani

June 9, 2021



Contents

1 Introduction

1.1 Physical model

In a classical microscopic model, a magnetic material is modelled by a lattice of classical spins $\mathbf{S}_i$ which are taken as unit vectors in $\mathbb{R}^3$. An adequate classical Hamiltonian H is introduced to take care of the interaction between spins and the external magnetic field. The Boltzmann distribution of such system is usually studied in order to determine the behaviour of the magnetic material at a defined temperature.

1.1.1 The Heisenberg Hamiltonian

The Heisenberg Hamiltonian is the simplest model for ferromagnetic and antiferromagnetic materials. In the most general form, the Hamiltonian is assumed to be:

$$H_H = -\frac{1}{2}\sum_{i,j} J_{ij}\mathbf{S}_i \cdot \mathbf{S}_j \tag{1}$$

In the simplest case we can assume nearest neighbour interaction with constant coupling everywhere,

$$J_{ij} = \begin{cases} J & \text{if neighbour} \\ 0 & \text{if not neighbour} \end{cases} \tag{2}$$

In this case the Hamiltonian is simplified to:

$$H_H = -J\sum_{\langle i,j\rangle} \mathbf{S}_i \cdot \mathbf{S}_j \tag{3}$$

where the $\langle i,j\rangle$ means sum on nearest neighbour. Hamiltonians with $J > 0$ models a ferromagnet while the case $J < 0$ represents an antiferromagnet. The classical Heisenberg model in 3 dimensions shows a continuous phase transition between the ordered phase (ferromagnetic phase) and the disordered phase (paramagnetic phase).

The Heisenberg model belongs to the greater family of the n-vector model, also known as $O(n)$, which is the family of all the system of interacting spins on a lattice where the spins is a unit vector in $\mathbb{R}^n$.

The name is justified as these Hamiltonians are invariant under transformations belonging to the orthogonal group $O(n)$.

Many useful systems belongs to this class:

- $O(1)$ is the classical Ising model
- $O(2)$ is the classical XY model
- $O(3)$ is the classical Heisenberg model
- $O(4)$ is a toy model for the Higgs sector of the Standard Model

Only $O(1)$ model is exactly solvable, for this reason computer simulations are extremely useful to understand the behaviour of these systems.

1.1.2 The 2D Heisenberg model

As stated before, the 3D Heisenberg model transition between ferromagnetic and paramagnetic phases is a continuous phase transition. Many new interesting materials under the technological point of view are bidimensional materials and therefore can be described with a 2D Heisenberg model. If this model is analyzed with analytical methods or computational physics tools, however, is curious to find that is not present any phase transition.

This interesting feature is explained by the Mermin-Wagner-Hohenberg Theorem which states that there is no phase with spontaneous breaking of a continuous symmetry for $T > 0$, in $d \leq 2$ dimensions.

1.1.3 Coupling with external magnetic field

The coupling with an external magnetic field is done with a Zeeman term which, after a suitable redefinition of $\mathbf{H}$, reads:

$$H_Z = -\sum_i \mathbf{H} \cdot \mathbf{S}_i \tag{4}$$

1.1.4 The antisymmetric interaction

Some interesting materials show a behaviour which can not be explained by a symmetric interaction only. The introduction of an antisymmetric term in the Hamiltonian give rise to different types of low temperature spontaneous ordered phases. This type of interactions, called Moriya-Dzyaloshinskii interaction (DMI) was introduced by Moriya in the article [1] and reads

$$H_{DM} = -\sum_{\langle i,j \rangle} \mathbf{D}_{ij} \cdot \mathbf{S}_i \times \mathbf{S}_j \tag{5}$$

The value of the parameter $_{ij}$ depends on the type of the interaction and is discussed in the original paper. For example the superexchange interaction can be modelled by a DMI with vector

$$\mathbf{D}_{ij} \propto \mathbf{r}_i \times \mathbf{r}_j \tag{6}$$

where $\mathbf{r}_i$ and $\mathbf{r}_i$ are the vectors of a third ion which mediates the interaction.

The full Hamiltionian with Heisenberg, DMI and Zeeman terms is:

$$H = H_H + H_{DM} + H_Z = -J \sum_{\langle i,j \rangle} \mathbf{S}_i \cdot \mathbf{S}_j - \sum_{\langle i,j \rangle} \mathbf{D}_{ij} \cdot \mathbf{S}_i \times \mathbf{S}_j - \sum_i \mathbf{H} \cdot \mathbf{S}_i \tag{7}$$

1.2 Simulation type and expected results

The aim of this project is to simulate some microscopic model of magnetic material and determine the features of the phase transitions in this model. In order to do so, a Monte Carlo simulation will be used with the Metropolis-Hastings algorithm.

Since the computational resources are very limited, is possible to simulate in reasonable time only cubic system with up to 20 spins per side. This limitation is far less hindering for the 2D systems simulations which can reach more than 64 spin per side. In all the simulations periodic boundary conditions will be adopted.

For each sample the total energy $\langle E_t \rangle$ and mean magnetization per spin $\langle \mathbf{m}_t \rangle$ will be recorded. For some simulations the spin distribution itself will be saved.

1.2.1 The Metropolis-Hastings algorithm

The Metropolis-Hastings algorithm is a Markov chain Monte Carlo method for obtaining a sequence of random samples from a probability distribution $P(x)$ (the target distribution) for which direct sampling is difficult and for which a function $f(x) \propto P(x)$ is known. The requirement that $f(x)$ has to be only be proportional to the density, rather than exactly equal to it, makes the Metropolis-Hastings algorithm particularly useful, because calculating the necessary normalization factor is often extremely difficult in practice in particular for statistical physics problems.

The Metropolis-Hastings algorithm works generating a Markov chain which distributions converges to $P(x)$. At each iteration, the algorithm picks a candidate c for the next sample value based on the current sample value x_t. Then, with some probability, the candidate is either accepted ($x_{t+1} = c$) or rejected ($x_{t+1} = x_t$) the probability of acceptance is determined by comparing the values of the function $f(x)$ of the current and candidate sample values.

The Metropolis algorithm works in the following way. Choose an arbitrary point x_0 to be the first sample, and choose an arbitrary probability density $g(x|y)$ that suggests a candidate for the next sample value x. For the Metropolis algorithm, g must be symmetric; in other words, it must satisfy $g(x|y) = g(y|x)$. A usual choice is to let $g(x|y)$ be a Gaussian distribution centered at y. The function g is referred to as the proposal density or jumping distribution.

After initialization, the Monte Carlo iteration is composed of the following steps

1. **Generation**
   Generate a candidate c for the next sample by picking from the distribution $g(c|x_t)$

2. **Calculation**
   Calculate the acceptance ratio $\alpha = f(c)/f(x_t)$ which will be used to decide whether to accept or reject the candidate. Because f is proportional to the density P, we have that $\alpha = f(c)/f(x_t) = P(c)/P(x_t)$

3. **Comparison**
   Generate a uniform random number u on $[0, 1]$. If $u \leq \alpha$ accept the candidate by setting $x_{t+1} = c$ If $u > \alpha$ reject the candidate and set $x_{t+1} = x_t$, instead.

1.2.2 Expected outcomes

In this project the aim is to try to replicate some results of [2] and [3] and study the thermodynamic properties of an Heisenberg ferromagnet in 3D.

Moreover, a set of simulations will be done for the 2D Heisenberg model to verify the absence of phase transitions.

Finally, an additional set of simulation will be performed to test the effect of the Dzyaloshinskii-Moriya interaction in 2D systems.

The expected outcomes of the simulations are:

1. Verify the alignment of spins in the ordered phase and show clearly the transition between ordered and disordered phase

2. Determine the critical temperature
3. Compute the spin-spin correlation function
4. Compute the critical exponents
5. Simulate an hysteresis cycle
6. Shows the absence of continuous phase transitions in 2D
7. Test the low temperature behaviour of a system which exhibits Dzyaloshinskii-Moriya interaction

2 Estimating the optimal separation of samples

The quality of the results of a Monte Carlo simulation depends on the correlation between the samples. In order to have an estimate of the correlation time, is useful to first make a trial simulation and measure the behaviour of the time correlation function defined as:

$$C_t(\tau) = \langle \mathbf{s}(\tau) \cdot \mathbf{s}(0) \rangle - \langle \mathbf{s}(\tau) \rangle \cdot \langle \mathbf{s}(0) \rangle \tag{8}$$

The averaging is taken on the ensemble and on the spatial coordinates. In this case, since we are dealing with a single cluster simulation, is not possible to have multiple samples for the same step number.

Simulations of a ferromagnetic cubic system with $L = 14, 16, 18, 20, 22$ spins per side have been performed with parameters $T = 2$ and $J = 1$ which corresponds to a disordered phase. The initial condition has been chosen as a totally random distribution and the simulation has run for 6×10^5 iterations taking samples every 500 Monte Carlo steps.

In figure 1 is shown the behaviour of the energy in the four simulations. The warm-up period has been chosen to be 2×10^5 steps. After that, the time correlation has been computed and is shown in figure 2.

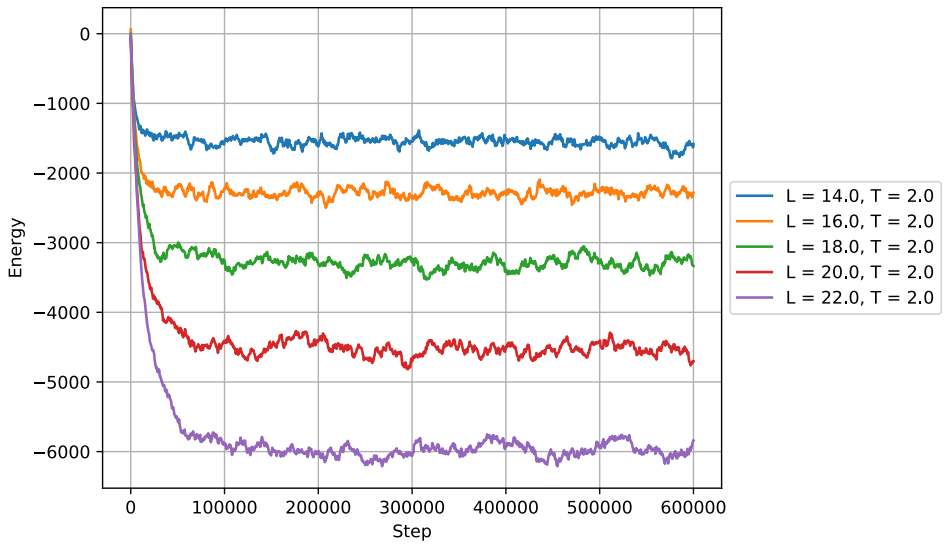


Figure 1: Energy in time correlation test simulation

Is possible to assume that the correlation is negligible after $\tau = 100 \times 10^3$ iteration and thus this number have been chosen as the number of iteration between samples, even if to have truly independent samples for any $L \leq 22$ would be needed at least $2\tau = 2 \times 10^4$ Monte Carlo steps in between two different measurement.

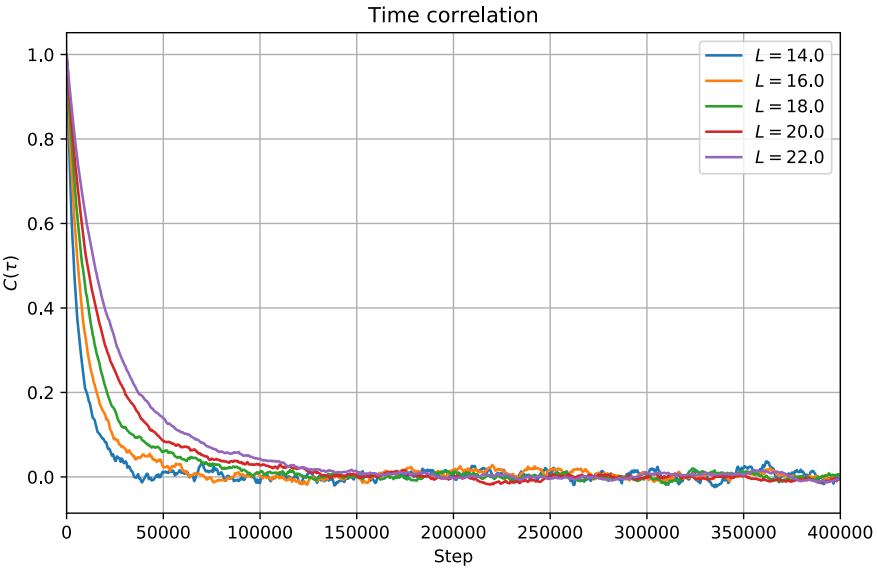


Figure 2: Time correlation

3 Simulation of a ferromagnetic material

In order to simulate the behaviour of a ferromagnetic material, simulations of simple cubic lattices of $L = 10, 12, 14, 16, 18, 20$ spins per side have been performed. The range of temperature chosen is $T \in [0.50, 2.50]$ taking measures every 0.1. To better understand the critical behaviour, points every 0.005 have been added in the region between $T = 1.4$ and $T = 1.5$ where the critical point is thought to be.

The true thermodynamic limit of a ferrmoagnetic system in the ordered phase is a single-domain structure. If we choose a random initial configuration, is more probable to reach a multi-domain phase which is quite stable over time. This situation should be avoided since, even in the end the final state will converge to a single cluster distribution, this will affect the quality of the simulation in the near-term. Even though, a totally aligned initial condition is not a good staring point, too, because is too stable. For this reason, the initial condition used is a partial-aligned spin configuration. This configuration of tilted spins is obtained generating samples from a Kent distribution with parameters $\kappa = 20$, $\beta = 0$, $\boldsymbol{\gamma}_1 = \mathbf{e}_z$.

For each system, 2.5×10^8 iterations have been performed taking samples every 100×10^3 Monte Carlo steps. A warm-up time of 1×10^7 steps has been proven to be enough to guarantee thermalized ensembles. It is known that the distribution of the energy of the system neglecting higher order correction is a normal distribution, with variance related to heat capacity of the system. Therefore a simple check on the distribution of energies of the samples can give information on the good behaviour of the algorithm. In fact, as can be seen in figure 3, the distribution is approximately Gaussian.

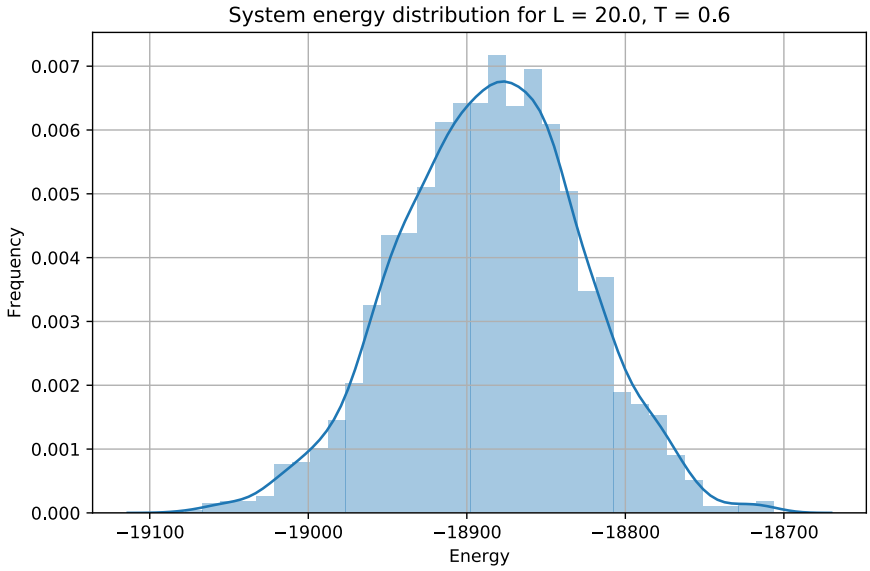


Figure 3: Distribution of energies of samples

As visible in figure number 4 and 5, the system energy $E_L(T)$ has a very smooth behaviour and the per site energies $e_L(t)$ converges perfectly, giving us hints on the trustworthiness of the simulation.

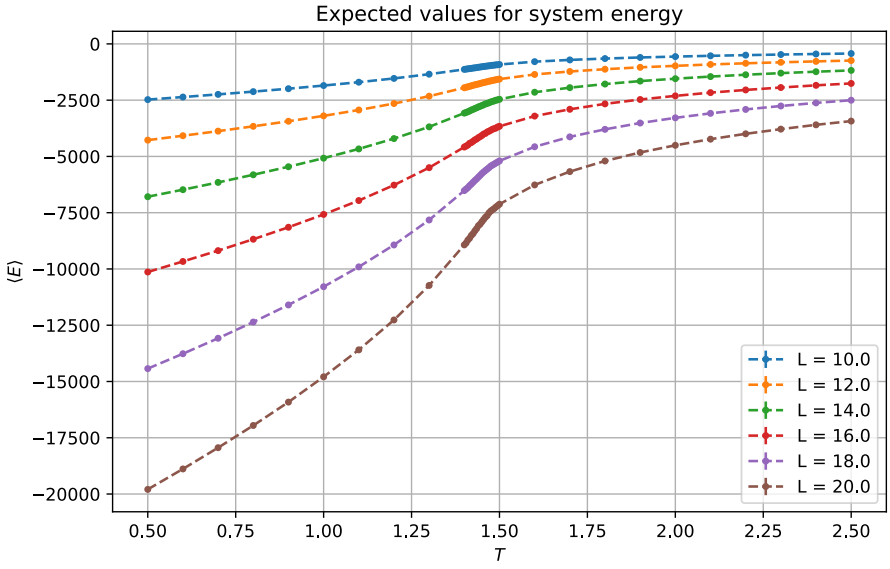


Figure 4: Energy of an Heisenberg ferromagnet

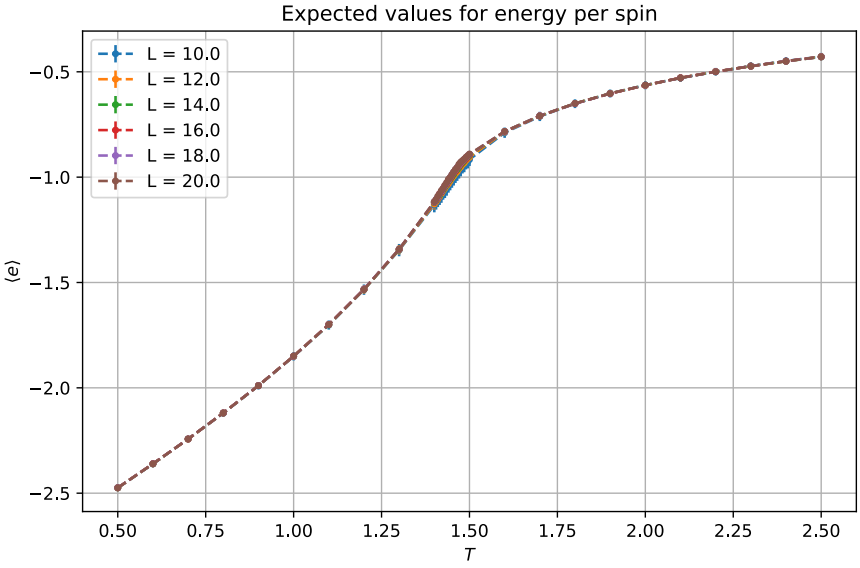


Figure 5: Energy per spin of an Heisenberg ferromagnet

From basic statistical physics is known that the a heat capacity estimator for spin system is

$$c_v(T) = \lim_{N\to+\infty} \frac{1}{N} \frac{\langle E^2\rangle - \langle E\rangle^2}{T} \tag{9}$$

where T is the temperature, $N = L^3$ is the number of spins and E are the samples energies (in all this report $k_B = 1$ is assumed).

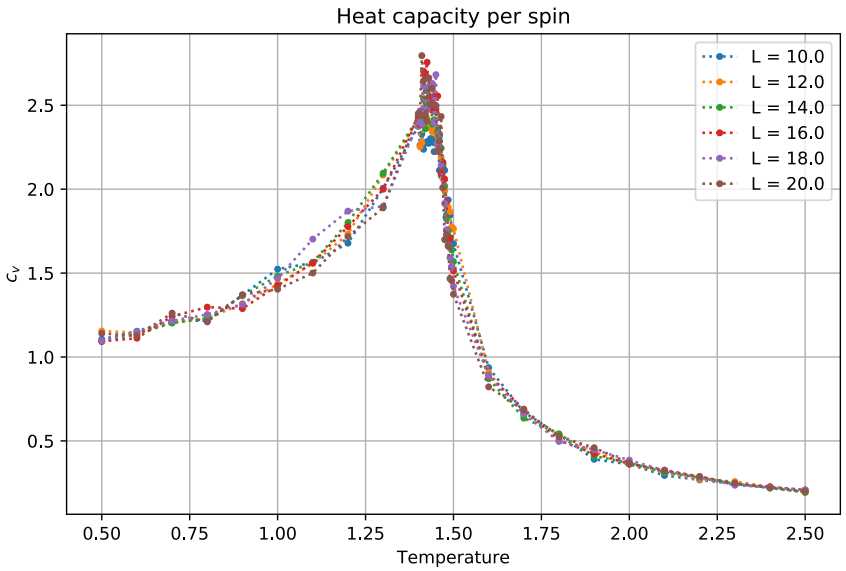


Figure 6: Heat capacity per spin in a Heisenberg ferromagnet

The heat capacity per spin as function of temperature $c_v(T)$ shows clearly a peak near $T = 1.4$ which confirm on the existence of a continuous phase transition and the guess on the critical temperature position. Also in this graph the lines for each system dimension shows a clear convergence as the system dimensions increase.

It worth notice as the heat capacity in the low temperature limits behaves like $\lim_{T\to 0} c_v(T) = 1$. This is due to equipartition theorem. In fact, the system supports spin waves each of one behaves as an oscillator (two degrees of freedom). The number of spin waves modes is equals that the number of lattice sites. Therefore this contribution is exactly

$$c_v = \frac{1}{N}\frac{\mathrm{d}E}{\mathrm{d}T} = \frac{1}{N}\frac{\mathrm{d}}{\mathrm{d}T}\frac{2N}{2}k_BT = 1 \tag{10}$$

From the figure 7 the second order nature of the phase transition is confirmed, as the order parameter $\mathbf{m}$ (per-spin magnetization) does not shows any finite jump. As the system size increase, the difference between the two phases became more marked.

An estimator for system susceptibility of a spin system is,

$$\chi_{i,j} = \lim_{N\to\infty} N\frac{\mathrm{Cov}(m_i, m_j)}{T} \tag{11}$$

where Cov is the statistical covariance and m_i are the per-spin magnetization vector component.

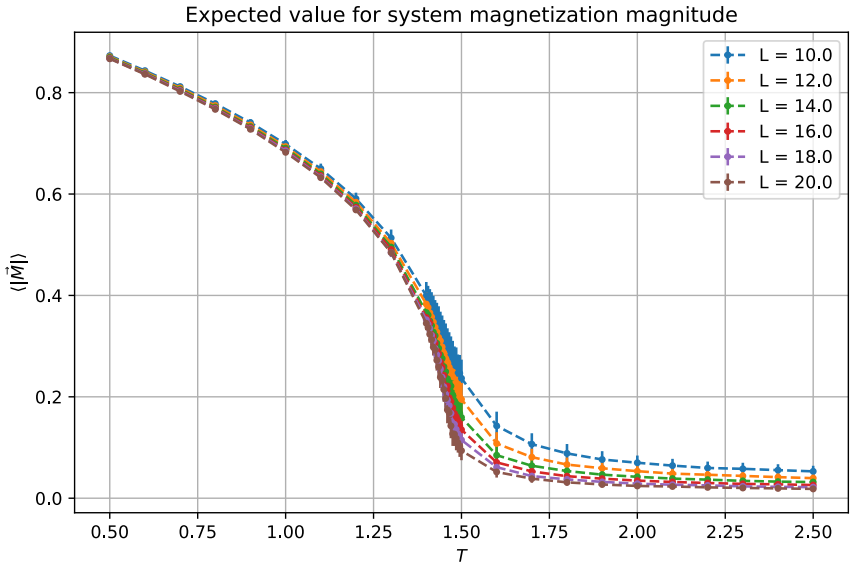


Figure 7: Mean magnetization magnitude for an Heisenberg ferromagnet

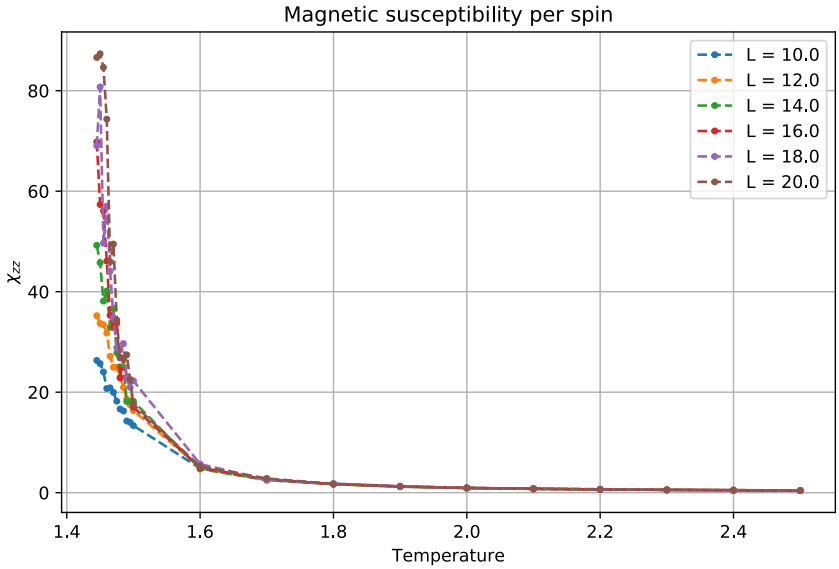


Figure 8: Magnetic susceptibility per spin along z-axis for the disordered phase of an Heisenberg magnet

If we plot the matrix component of the susceptibility tensor as in figure 9, is possible to notice as the anisotropic components rapidly became zero in the disordered phase.

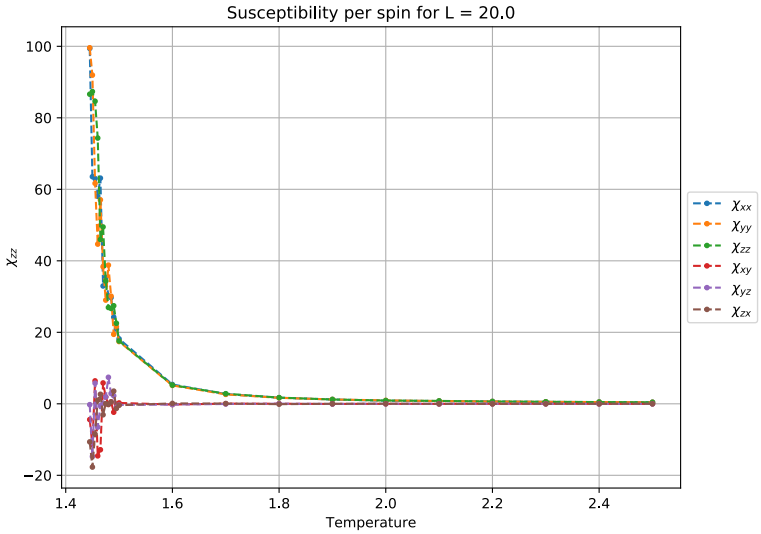


Figure 9: Magnetic susceptibility per spin for the disordered phase of an Heisenberg magnet with $L = 20$

3.1 Critical temperature

In order to precisely determine the critical temperature, the fourth order cumulant has been used.

$$U_L = 1 - \frac{1}{3}\frac{\langle m^4 \rangle}{\langle m^2 \rangle^2} \tag{12}$$

Is it clear that in the ordered phase $U_L \longrightarrow 2/3$ while in the disordered one $U_L \longrightarrow 4/9$. Is well know that, neglecting higher order corrections, the lines of $U_L(T)$ for various system sizes cross at the point (T_c, U_L^*). Therefore checking the intersection of these lines is a powerful tool to determine the critical temperature. In figure 10 the cumulant lines are plotted against temperature, a zoomed picture around $T = 1.45$ is shown in figure 11.

Carefully analyzing the crossing point, is clear that the transition temperature is near $T = 1.445$ and therefore this value will be used in the next analysis.

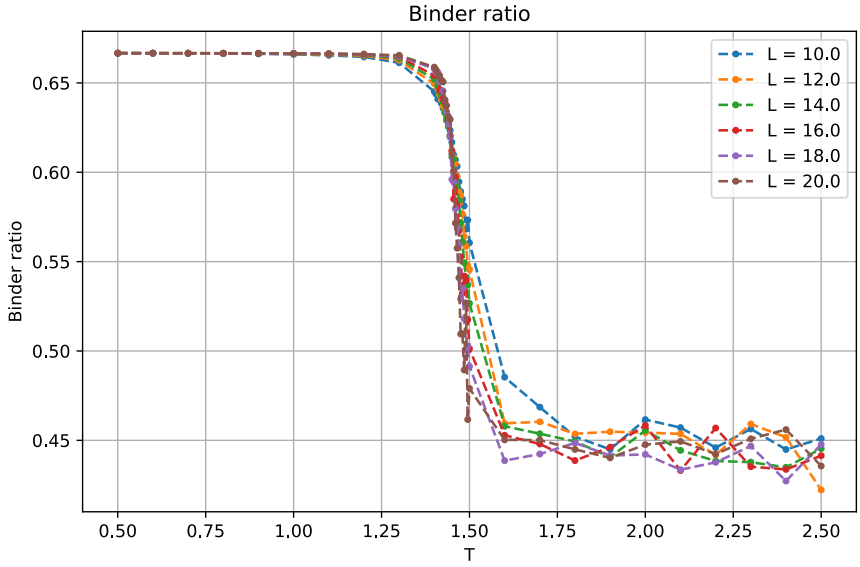


Figure 10: Binder ratio for an Heisenberg magnet

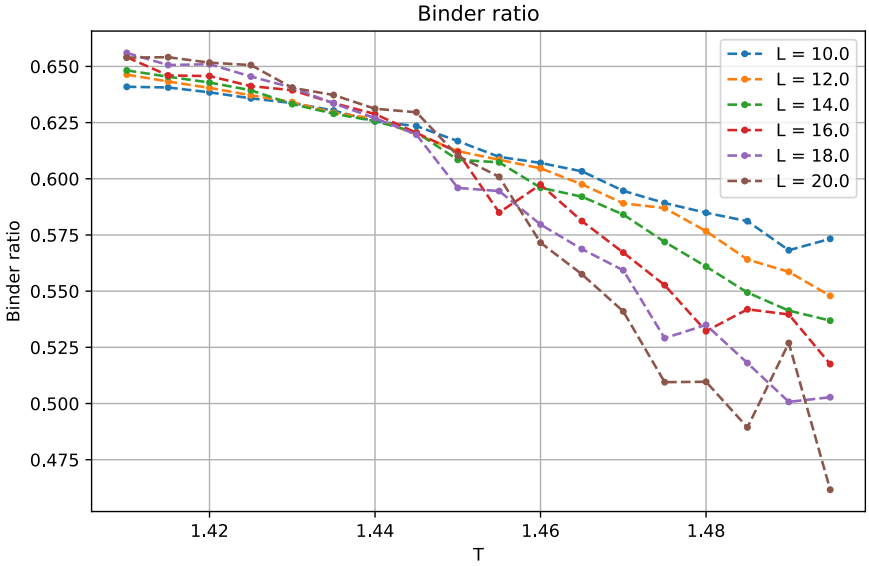


Figure 11: Binder ratio for an Heisenberg magnet, zoomed

3.2 Critical exponents

It is known that at the critical temperature the magnetization scales like

$$\langle m \rangle \propto L^{-\beta/\nu} \tag{13}$$

Therefore taking the natural logarithm $\ln\langle m \rangle = c - \frac{\beta}{\nu} \ln L$. A linear regression has thus been performed to estimate the ratio between the two critical indices. The resulting estimate is $\frac{\beta}{\nu} = 0.548(1)$.

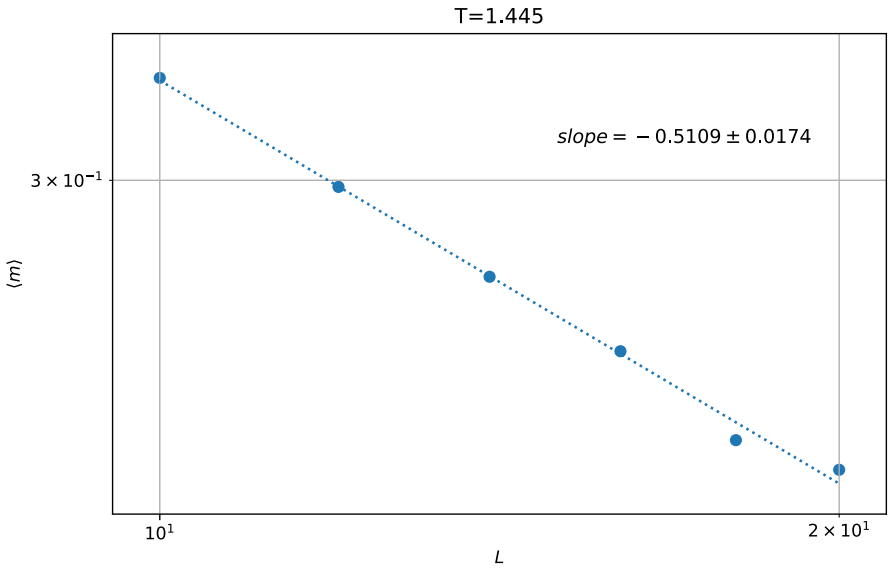


Figure 12: Log-Log plot of $\langle m \rangle$ vs L with linear regression for an Heisenberg ferromagnet

The same procedure can be used to derive γ/ν because the susceptibility scales like

$$\chi \propto L^{\gamma/\nu} \tag{14}$$

Therefore, always using linear regression is possible to estimate $\gamma/\nu = 1.9(2)$
For the estimate of the ν critical exponent, is possible to use the scale relation

$$\frac{\mathrm{d}U_L}{\mathrm{d}(T^{-1})} \propto L^{1/\nu} \tag{15}$$

A smart way to obtain the derivative of the fourth order cumulant with respect of the inverse temperature is to find a thermodynamic estimate of such quantity. It can be proved that a suitable formula is the following

$$\frac{\mathrm{d}U_L}{\mathrm{d}(T^{-1})} = (1 - U_L)\left[\langle E \rangle - 2\frac{\langle m^2 E \rangle}{\langle m^2 \rangle} + \frac{\langle m^4 E \rangle}{\langle m^4 \rangle}\right] \tag{16}$$

This make possible to avoid noisy numerical derivatives. The result of the estimate is shown in figure 14.

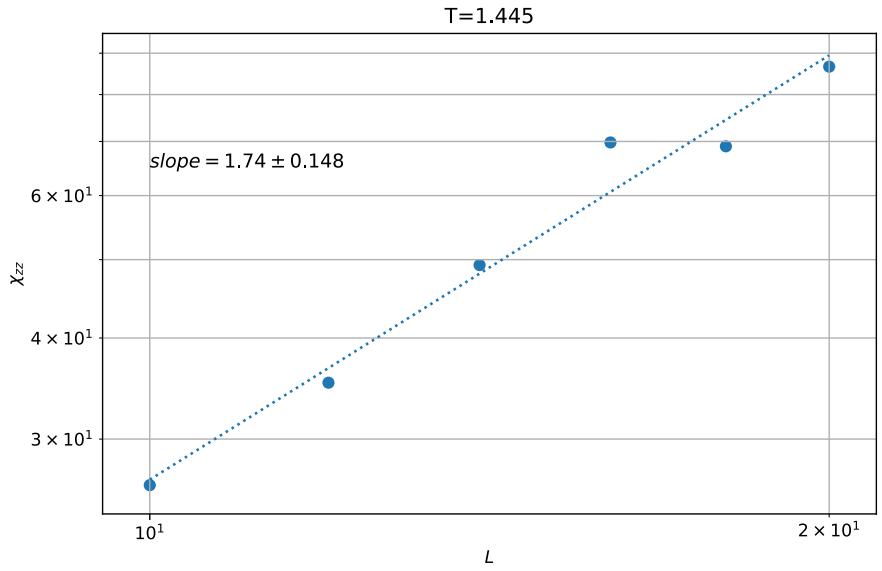


Figure 13: Log-Log plot of χ vs L with linear regression for an Heisenberg ferromagnet

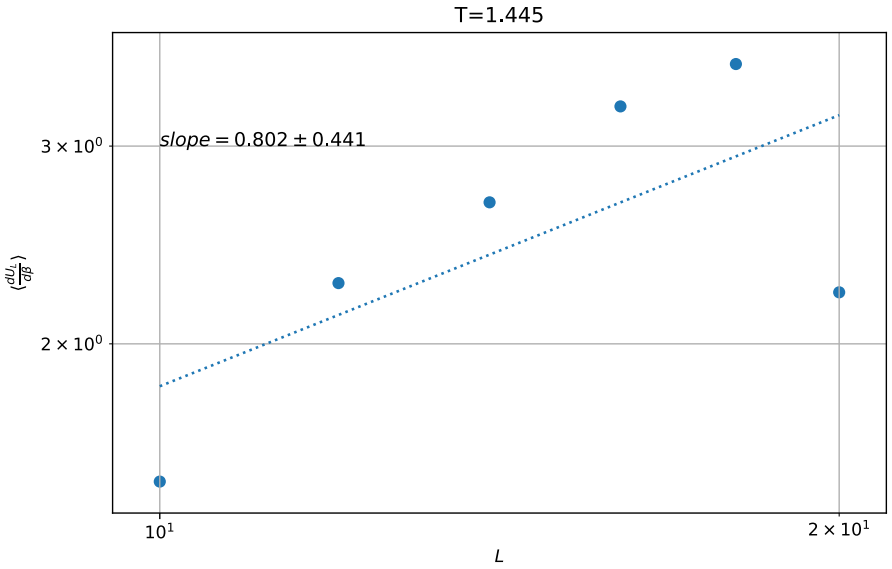


Figure 14: Log-Log plot of $\frac{\mathrm{d}U_L}{\mathrm{d}(T^{-1})}$ vs L with linear regression for an Heisenberg ferromagnet

From the estimate of β/ν, γ/ν and ν is we can directly determine β and γ. We can determine other critical exponents using the well know relation between critical exponents such as the hyperscaling relation,

$$\alpha = 2 - d\nu \tag{17}$$

Widom's law,

$$\delta = 1 + \gamma/\beta \tag{18}$$

and finally

$$\eta = 2 - \gamma/\nu \tag{19}$$

The results of the estimates are collected in the table 1 where are compared to the values obtained by analytic methods (ϵ-expansion).

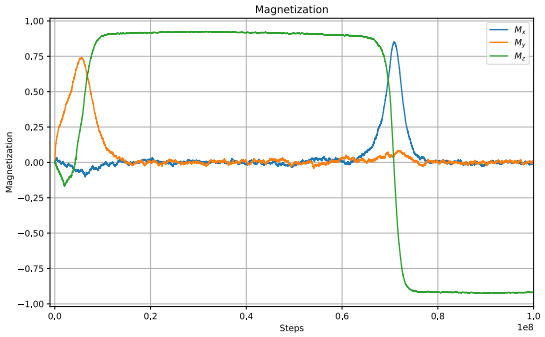
Critical index	This project estimate	ϵ-expansion
ν	0.80(44)	0.705(3)
γ/ν	1.74(15)	1.966(14)
γ	1.40(77)	1.386(4)
β/ν	0.511(17)	0.517(6)
β	0.41(22)	0.3645(25)
α	$-0.4(13)$	$-0.115(9)$
δ	4.41(40)	4.802(37)
η	0.26(15)	0.040(3)

Table 1: Critical indices

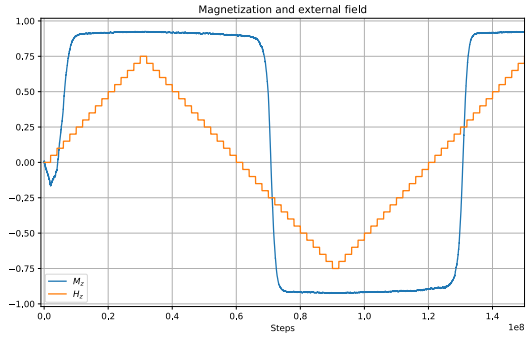
As can be notice, most of the estimate are within one standard deviation from the most precise value. It is clear that the main source of errors is the estimate of ν as the linear regression does not fit very well the point obtained. This is due to the complex formula used to estimate the Binder ratio derivative which leverage fourth order moment of the distribution and thus probably have longer convergence times.

3.3 Simulation of an hysteresis cycle

To verify if the model is able to reproduce an hysteresis cycle, a simulation with a dynamically varying external field along the z axis has been performed. Since the expected outcome is only a qualitative plot of the behaviour, the distance between two sample has been selected to be only 5×10^3 iterations with random initial spins orientations. The external field H_z vary from 0 to ± 0.75 while the temperature chosen is $T = 0.4$ in order to have a stable ordered phase. The results are plotted below.



(a) Magnetization per spin components as function of sample number



(b) Magnetization per spin along z and external field as function of sample number

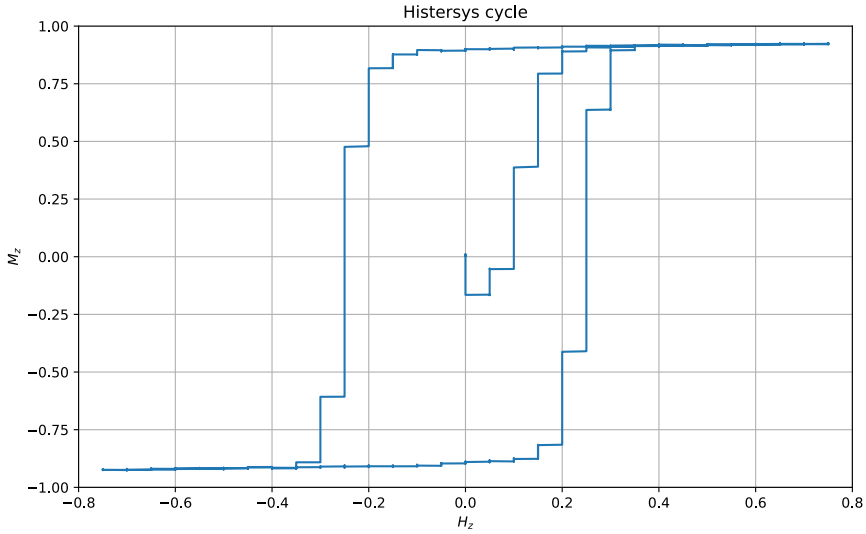


Figure 16: Hysteresis cycle

The results are good, except for the inevitable spontaneous magnetization at the beginning of the simulation. A very high residual magnetization can be identified and the coercive field is about 0.25.

4 Simulation of a 2D material

In this section the simulation of the Heisenberg model of a 2D material will be analyzed. Obviously the time correlations became negligible far faster than in the 3D system. This led to the choice of a separation of 5×10^3 Monte Carlo steps between samples.

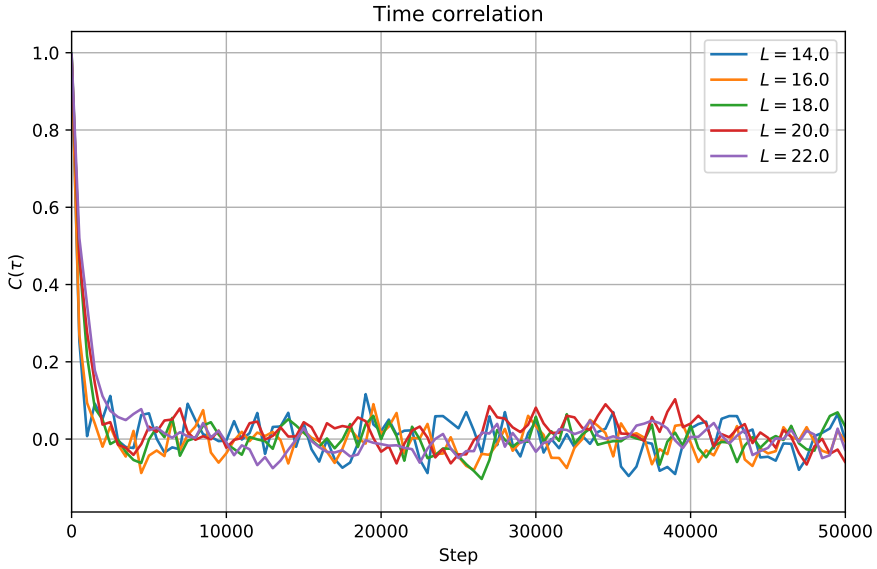
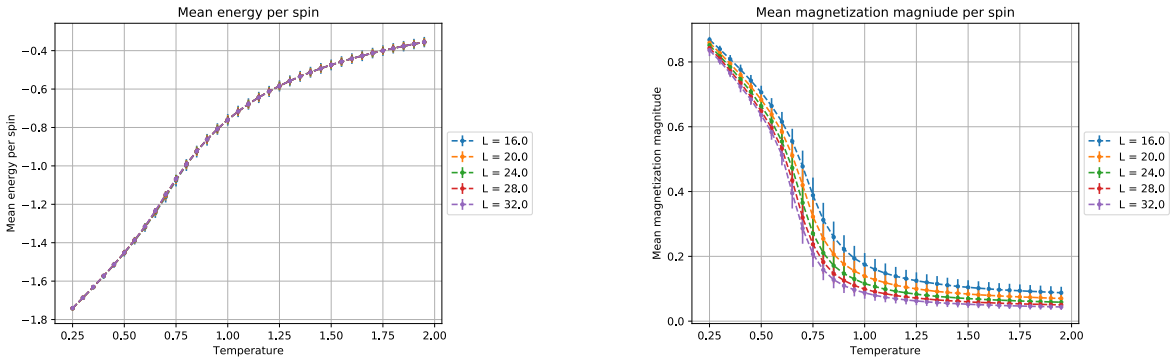


Figure 17: Time correlation of a 2D Heisenberg ferromagnet

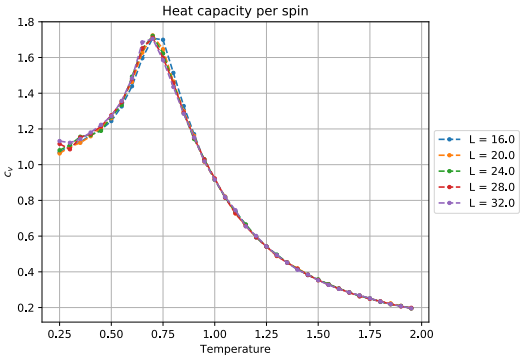


(a) Energy per spin for a 2D Heisenberg ferromagnet (b) Magnetization magnitude per spin for a 2D Heisenberg ferromagnet

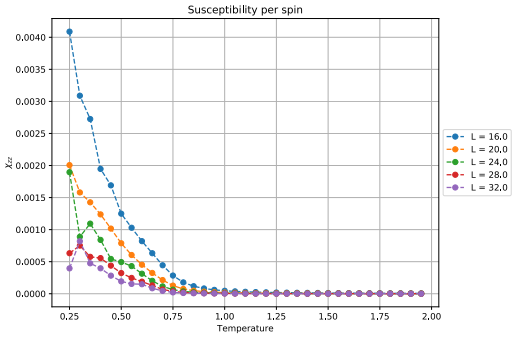
Figure 18: 2D Heisenberg ferromagnet

As can be see by figure 18b, the system exhibits a magnetized phase at low temperatures.

The analysis of the heat capacity in figure 19a shows clearly a peak around $T = 0.7$, however is important to notice that it seems there is not a discontinuity here. Therefore it cannot be considered a phase transition but we can talk of a pseudo-critical behaviour.



(a) Heat capacity per spin for a 2D Heisenberg ferromagnet



(b) System susceptibility per spin for a 2D Heisenberg ferromagnet

Figure 19: 2D Heisenberg ferromagnet

If we observe the susceptibility in figure 19b this guess is confirmed as there is not any divergence in this region.

The Binder ratios curves in figure 20 does not seem to have an intersection in the transition region.

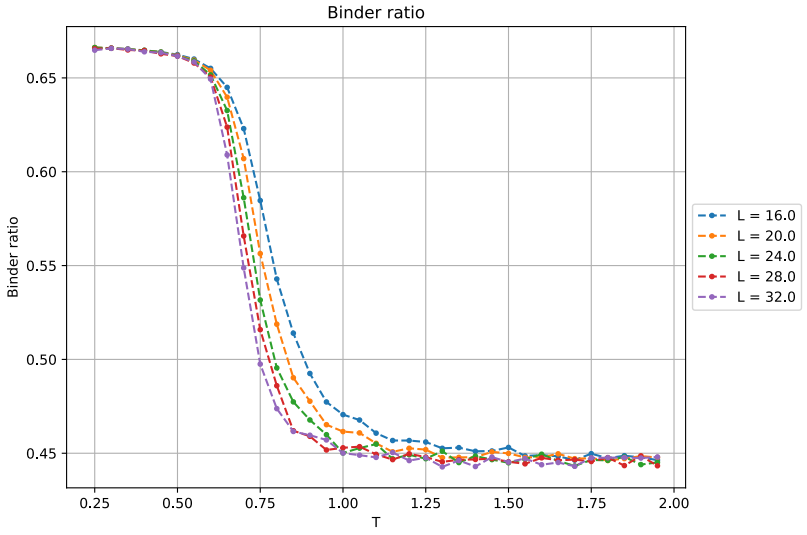


Figure 20: Binder ratio per spin for a 2D Heisenberg ferromagnet

The current thinking is that there is not a true phase transition, but instead it exhibits pseudocritical behaviour: the correlation length becomes extremely large, but not infinite. The latest publication on the topic is [4] where a finite-size-scaling analysis was employed on systems up to 1024x1024 In this paper, the XY model is also studied and compared with the Heisenberg model. The conclusion seems to be that they are different but it is very difficult to show the difference un-

less considerable computational effort is spent, and a complicated finite size analysis is conducted. In both cases a peak in the heat capacity curve is seen, and a pseudo-critical temperature can be identified.

However, the definition of phase transition imply that the correlation length scales in a systematic way as the system size becomes very large. For the XY model, this seems to be the case while for the Heisenberg model, it seems not.

Moreover, while a BKT phase transition for the XY model can be explained in terms of topological defects in the configurations of spins confined to the plane, the Heisenberg model does not have these well-defined defects. So a BKT transition is not expected for the Heisenberg model. Nonetheless, the system has been known for years to exhibit behaviour looking like a phase transition.

4.1 Material with antisymmetric interaction

In this section some test simulations of an antisymmetric 2D material will be performed used a DMI with vector

$$\mathbf{D}_{ij} = D\mathbf{r}_{ij} \tag{20}$$

where $\mathbf{r}_{ij}$ is a vector which connects the two sites.

The full Hamiltonian with Heisenberg, DMI and Zeeman terms is:

$$H = -J\sum_{\langle i,j\rangle}\mathbf{S}_i\cdot\mathbf{S}_j - D\sum_{\langle i,j\rangle}\mathbf{r}_{ij}\cdot\mathbf{S}_i\times\mathbf{S}_j - \sum_i \mathbf{H}\cdot\mathbf{S}_i \tag{21}$$

In a first run of simulations, lattice with $L = 24, 32, 40, 48, 56, 64$ spins per side have been simulated with parameters $J = 1$ and $D = 1$ for temperatures $T = 0.2, 0.3, \ldots 3$.

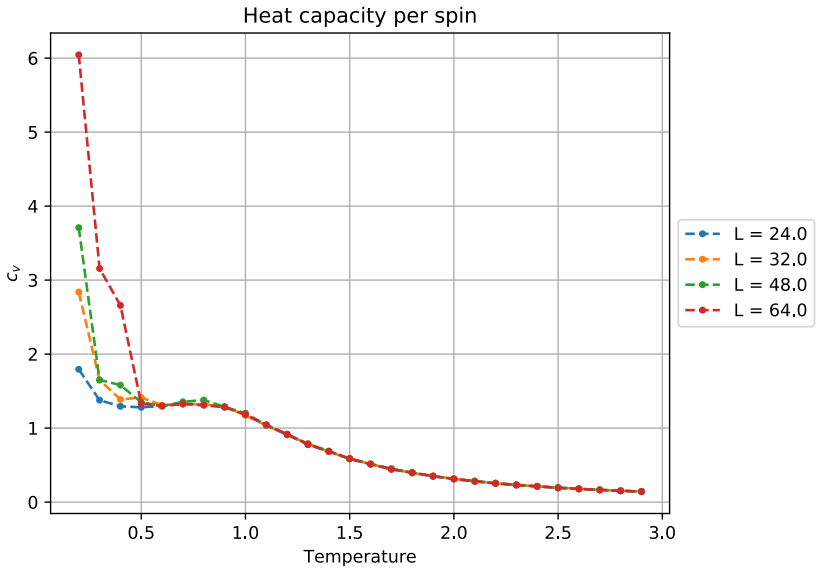


Figure 21: Heat capacity per spin for a system with DMI

The curves of the heat capacities seem to diverge in the low temperature region, but they nicely converge starting from $T = 0.5$.

The susceptibility plot shows a trough below $T = 0.5$ suggesting something interesting is happening in the low temperature region. However, since these chart does not show a very clear behavior, next will be showed some quiver plots. The HLS colormap is used because is more adequate for 2D plots. In particular, Hue is determined by the direction in the xy-plane and the Litheness by the orientation along z.

In the figure 24 we can observe the existence of a low temperature ordered phase made of long "chain" of spins aligned toward the same direction. This kind of structure seems to exists up to $T = 0.5$ then there is a transition up to a disordered phase for $T > 0.8$, see figures 25 and 26.

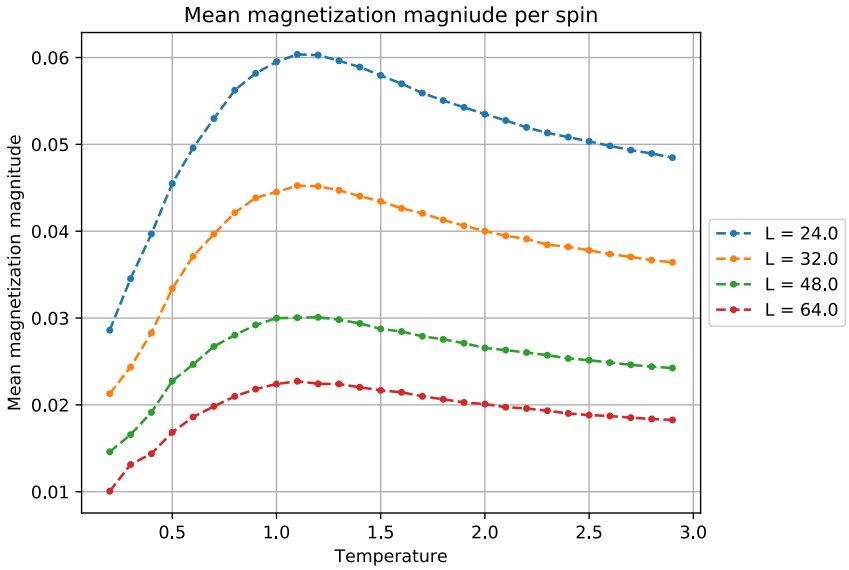


Figure 22: Magnetization magnitude for a system with DMI

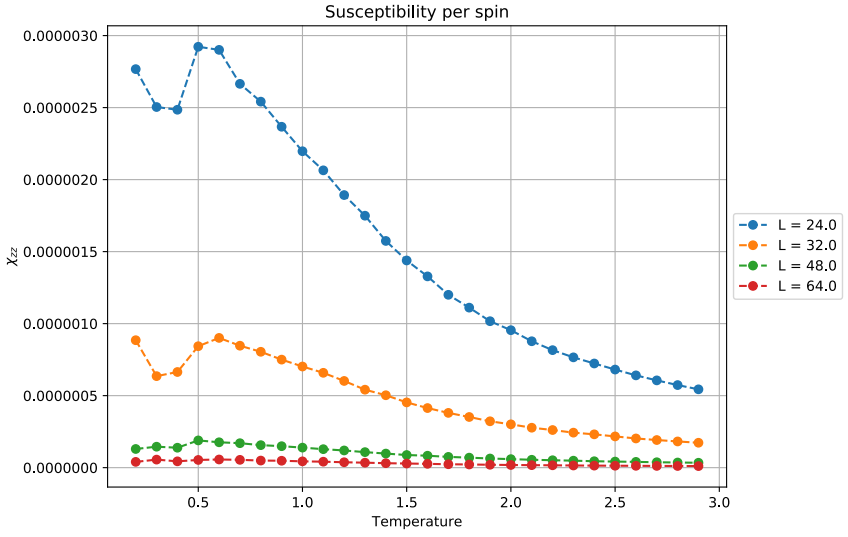


Figure 23: Susceptibility per spin for a system with DMI

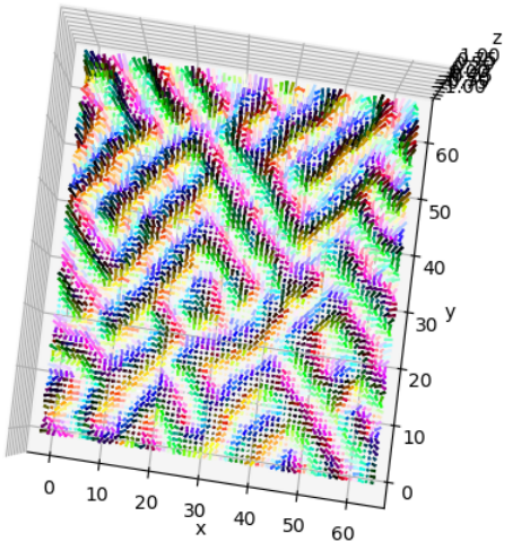


Figure 24: Low temperature ordered phase, $T = 0.3$

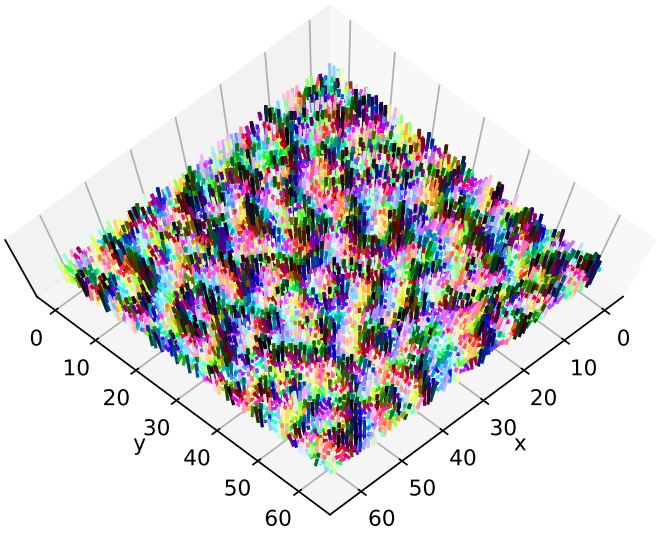


Figure 25: Transition phase, $T = 0.7$

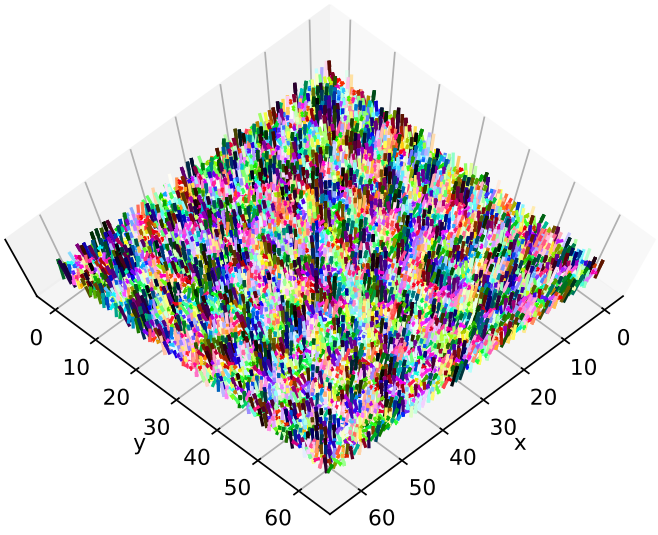


Figure 26: Disordered phase, $T = 1$

4.1.1 External magnetic field

In figure 27 the same system have been perturbed with an external magnetic field $\mathbf{H} = 0.5\mathbf{e}_z$ at temperature $T = 0.2$. The chains are broken by the external field and some globular formation appears forming what seems to be a lattice of magnetic skyrmions.

(a) Low temperature perturbed state (b) Low temperature perturbed spin distribution

Figure 27: Skyrmion lattice, $T = 0.2$, $H = 0.6$

(a) Transition perturbed state (b) Transition perturbed spin distribution

Figure 28: Skyrmion lattice transition, $T = 0.6$, $H = 0.6$

As can be seen in figure 28 and 29, this phase is stable up to $T = 0.5$.

(a) High temperature perturbed state (b) High temperature perturbed spin distribution

Figure 29: Disordered perturbed state at high temperature, $T = 0.9$, $H = 0.6$

5 Technical details of the implementation

The program is written in Python 3.7 using NumPy library for linear algebra. The Numba python package is used to produce JIT compiled code for the performance critical part of the program. To speed up the computation, a multiprocess architecture has been adopted to run a simulation on each core of the CPU. The results of the simulations are analyzed using Jupyter notebooks.

The program has been tested both in a version which use polar coordinates for which the spin state was stored like $S_i = (\theta, \phi)$ and in one using cartesian coordinates $S_i = (x, y, z)$. Unexpectedly, for the simple Heisenberg Hamiltonian the version using polar coordinates shows better performance (up to 10% faster execution) and thus has been used for the majority of the simulations. However, the generalization to more complex interactions is difficult in this coordinate system and would probably show a worse performance due to the high number of evaluations of trigonometric functions needed.

All the source code is available on `https://github.com/skdys/thermalspin` under MIT License.

In the following lines, the main file with the core code of the simulation will be included.

Listing 1: spin_system.py

```
#!/usr/bin/env python3

"""
Classical Spin System Monte Carlo simulator
"""

import numpy as np
from numba import jit

from skdylib.spherical_coordinates import xyz_sph_urand

USE_NUMBA = True


class SpinSystem:
    """
    This class represent a spin system with Heisenberg interaction, DMI and Zeeman terms
    """

    def __init__(self, initial_state, J, D, Hz, T):
        self.J = J
        self.D = D

        self.Hz = Hz
        self.T = T
        self.beta = 1 / T

        self.state = initial_state
        self.nx = self.state.shape[0]
        self.ny = self.state.shape[1]
        self.nz = self.state.shape[2]
        self.sites_number = self.nx * self.ny * self.nz

```

```
34          # Set the best energy functions depending on the the presence of an
                antisymmetric term
35          self.step_function = compute_step_Heisenberg if D == 0 else compute_step_DMI
36          self.compute_energy_function = compute_energy_Heisenberg if D == 0 else
                compute_energy_DMI
37
38          # Compute energy and magnetization of the initial initial_state
39          self.energy = self.compute_energy_function(self.state, self.nx, self.ny,
                self.nz, J, D, Hz)
40          self.total_magnetization = compute_magnetization(self.state)
41
42      @property
43      def magnetization(self):
44          """
45          The magnetization of the system
46          :return: The value of the magnetization
47          """
48          return self.total_magnetization / self.sites_number
49
50      def step(self):
51          """
52          Evolve the system computing a step of Metropolis-Hastings Monte Carlo.
53          It actually calls the non-object oriented procedure.
54          """
55          s, e, m = self.step_function(self.state, self.nx, self.ny, self.nz, self.J,
                self.D, self.Hz, self.beta,
56                                       self.energy,
57                                       self.total_magnetization)
58          self.state = s
59          self.energy = e
60          self.total_magnetization = m
61
62
63  # Compiled functions
64
65  @jit(nopython=USE_NUMBA, cache=USE_NUMBA)
66  def compute_magnetization(state):
67      """
68      Compute the total magnetization
69      :return: [Mx, My, Mz] vector of mean magnetization
70      """
71      mx = np.sum(state[:,:,:,0])
72      my = np.sum(state[:,:,:,1])
73      mz = np.sum(state[:,:,:,2])
74      return np.array([mx,my,mz])
75
76
77  # Pure Heisenberg model
78
79  @jit(nopython=USE_NUMBA, cache=USE_NUMBA)
80  def neighbhour_energy_Heisenberg(i, j, k, ii, jj, kk, state, J):
81      """
82      Compute the energy of two adjacent spins due to the Heisenberg Hamiltonian
83      :return: the energy computed
84      """
85      heisenberg_term = -J*np.dot(state[i, j, k], state[ii, jj, kk])
```

```
86      return heisenberg_term
87
88  @jit(nopython=USE_NUMBA, cache=USE_NUMBA)
89  def compute_energy_Heisenberg(state, nx, ny, nz, J, D, Hz):
90      """
91      Compute the energy of the system with Heisenberg Hamiltonian
92      :return: The value of the energy
93      """
94
95      energy_counter = 0.0
96
97      for i, j, k in np.ndindex(nx, ny, nz):
98          ii = (i + 1) % nx
99          energy_counter += neighbhour_energy_Heisenberg(i, j, k, ii, j, k, state, J)
100
101         jj = (j + 1) % ny
102         energy_counter += neighbhour_energy_Heisenberg(i, j, k, i, jj, k, state, J)
103
104         if nz > 1:
105             kk = (k + 1) % nz
106             energy_counter += neighbhour_energy_Heisenberg(i, j, k, i, j, kk, state,
                    J)
107
108         energy_counter += - Hz * state[i, j, k, 2]
109
110     return energy_counter
111
112
113 @jit(nopython=USE_NUMBA, cache=USE_NUMBA)
114 def compute_step_Heisenberg(state, nx, ny, nz, J, D, Hz, beta, energy,
    total_magnetization):
115     """
116     Evolve the system computing a step of Metropolis-Hastings Monte Carlo.
117     This non OOP function is accelerated trough jit compilation.
118     """
119
120     # Select a random spin in the system
121     i = np.random.randint(0, nx)
122     j = np.random.randint(0, ny)
123     k = np.random.randint(0, nz)
124
125     # Compute the energy due to that spin
126     e0 = 0
127
128     ii = (i + 1) % nx
129     e0 += neighbhour_energy_Heisenberg(i, j, k, ii, j, k, state, J)
130
131     ii = (i - 1) % nx
132     e0 += neighbhour_energy_Heisenberg(i, j, k, ii, j, k, state, J)
133
134     jj = (j + 1) % ny
135     e0 += neighbhour_energy_Heisenberg(i, j, k, i, jj, k, state, J)
136
137     jj = (j - 1) % ny
138     e0 += neighbhour_energy_Heisenberg(i, j, k, i, jj, k, state, J)
139
```

```
    if nz > 1:
        kk = (k + 1) % nz
        e0 += neighbhour_energy_Heisenberg(i, j, k, i, j, kk, state, J)

        kk = (k - 1) % nz
        e0 += neighbhour_energy_Heisenberg(i, j, k, i, j, kk, state, J)

    e0 += -Hz * state[i, j, k, 2]

    # Generate a new random direction and compute energy due to the spin in the new direction
    old_spin = state[i, j, k].copy()
    state[i, j, k] = xyz_sph_urand()

    e1 = 0

    ii = (i + 1) % nx
    e1 += neighbhour_energy_Heisenberg(i, j, k, ii, j, k, state, J)

    ii = (i - 1) % nx
    e1 += neighbhour_energy_Heisenberg(i, j, k, ii, j, k, state, J)

    jj = (j + 1) % ny
    e1 += neighbhour_energy_Heisenberg(i, j, k, i, jj, k, state, J)

    jj = (j - 1) % ny
    e1 += neighbhour_energy_Heisenberg(i, j, k, i, jj, k, state, J)

    if nz > 1:
        kk = (k + 1) % nz
        e1 += neighbhour_energy_Heisenberg(i, j, k, i, j, kk, state, J)

        kk = (k - 1) % nz
        e1 += neighbhour_energy_Heisenberg(i, j, k, i, j, kk, state, J)

    e1 += -Hz * state[i, j, k, 2]


    # Apply Metropolis algorithm
    w = np.exp(beta * (e0 - e1))
    dice = np.random.uniform(0, 1)

    if dice < w:
        energy += (e1 - e0)
        total_magnetization += state[i, j, k] - old_spin

    else:
        state[i,j,k] = old_spin

    return state, energy, total_magnetization


# Heisenberg interaction + DMI

@jit(nopython=USE_NUMBA, cache=USE_NUMBA)
def compute_energy_DMI(state, nx, ny, nz, J, D, Hz):
```

```
    """
    Compute the energy of the system with Heisenberg, DMI and Zeeman term
    :return: The value of the energy
    """

    energy_counter = 0.0

    Axp = np.array(([-J, 0, 0],
                    [0, -J, -D],
                    [0, +D, -J]))

    Ayp = np.array(([-J, 0, +D],
                    [0, -J, 0],
                    [-D, 0, -J]))

    Azp = np.array(([-J, -D, 0],
                    [+D, -J, 0],
                    [0, 0, -J]))

    for i, j, k in np.ndindex(nx, ny, nz):
        ii = (i + 1) % nx
        energy_counter += state[i, j, k].T.dot(Axp).dot(state[ii, j, k])

        jj = (j + 1) % ny
        energy_counter += state[i, j, k].T.dot(Ayp).dot(state[i, jj, k])

        if nz > 1:
            kk = (k + 1) % nz
            energy_counter += state[i, j, k].T.dot(Azp).dot(state[i, j, kk])

        energy_counter += - Hz * state[i, j, k, 2]

    return energy_counter


@jit(nopython=USE_NUMBA, cache=USE_NUMBA)
def compute_step_DMI(state, nx, ny, nz, J, D, Hz, beta, energy, total_magnetization):
    """
    Evolve the system computing a step of Metropolis-Hastings Monte Carlo.
    """

    Axp = np.array(([-J, 0, 0],
                    [0, -J, -D],
                    [0, +D, -J]))

    Axn = np.array(([-J, 0, 0],
                    [0, -J, +D],
                    [0, -D, -J]))

    Ayp = np.array(([-J, 0, +D],
                    [0, -J, 0],
                    [-D, 0, -J]))

    Ayn = np.array(([-J, 0, -D],
                    [0, -J, 0],
```

```
                    [+D,  0, -J]))

    Azp = np.array(([-J, -D,  0],
                    [+D, -J,  0],
                    [0,  0, -J]))

    Azn = np.array(([-J, +D,  0],
                    [-D, -J,  0],
                    [0,  0, -J]))

    # Select a random spin in the system
    i = np.random.randint(0, nx)
    j = np.random.randint(0, ny)
    k = np.random.randint(0, nz)

    # Compute the energy due to that spin
    e0 = 0

    ii = (i + 1) % nx
    e0 += state[i, j, k].T.dot(Axp).dot(state[ii, j, k])

    ii = (i - 1) % nx
    e0 += state[i, j, k].T.dot(Axn).dot(state[ii, j, k])

    jj = (j + 1) % ny
    e0 += state[i, j, k].T.dot(Ayp).dot(state[i, jj, k])

    jj = (j - 1) % ny
    e0 += state[i, j, k].T.dot(Ayn).dot(state[i, jj, k])

    if nz > 1:
        kk = (k + 1) % nz
        e0 += state[i, j, k].T.dot(Azp).dot(state[i, j, kk])

        kk = (k - 1) % nz
        e0 += state[i, j, k].T.dot(Azn).dot(state[i, j, kk])

    e0 += -Hz * state[i, j, k, 2]

    # Generate a new random direction and compute energy due to the spin in the new
        direction
    old_spin = state[i, j, k].copy()
    state[i, j, k] = xyz_sph_urand()

    e1 = 0

    ii = (i + 1) % nx
    e1 += state[i, j, k].T.dot(Axp).dot(state[ii, j, k])

    ii = (i - 1) % nx
    e1 += state[i, j, k].T.dot(Axn).dot(state[ii, j, k])

    jj = (j + 1) % ny
    e1 += state[i, j, k].T.dot(Ayp).dot(state[i, jj, k])

    jj = (j - 1) % ny
```

```
    e1 += state[i, j, k].T.dot(Ayn).dot(state[i, jj, k])

    if nz > 1:
        kk = (k + 1) % nz
        e1 += state[i, j, k].T.dot(Azp).dot(state[i, j, kk])

        kk = (k - 1) % nz
        e1 += state[i, j, k].T.dot(Azn).dot(state[i, j, kk])

    e1 += -Hz * state[i, j, k, 2]

    # Apply Metropolis algorithm
    w = np.exp(beta * (e0 - e1))
    dice = np.random.uniform(0, 1)

    if dice < w:
        energy += (e1 - e0)
        total_magnetization += state[i, j, k] - old_spin

    else:
        state[i, j, k] = old_spin

    return state, energy, total_magnetization
```

References

[1] Tôru Moriya. Anisotropic superexchange interaction and weak ferromagnetism. *Phys. Rev.*, 120:91–98, Oct 1960.

[2] Christian Holm and Wolfhard Janke. Critical exponents of the classical three-dimensional heisenberg model: A single-cluster monte carlo study. *Physical Review B*, 48(2):936950, Jan 1993.

[3] P. Peczak, Alan M. Ferrenberg, and D. P. Landau. High-accuracy monte carlo study of the three-dimensional classical heisenberg ferromagnet. *Physical Review B*, 43(7):60876093, Jan 1991.

[4] Yusuke Tomita. Finite-size scaling analysis of pseudocritical region in two-dimensional continuous-spin systems. *Phys. Rev. E*, 90:032109, Sep 2014.

[5] M. E. J. Newman and G. T. Barkema. *Monte Carlo Methods in Statistical Physics*. Clarendon Press, 1999.