Heisenberg Model in 3D

Mateo Cárdenes Wuttig, Justus Grabowsky, Daniel Schwarzenbach, Constança Tropa

June 7, 2025

Abstract

Our implementation of the Metropolis and Wolff algorithms in C++ offers a comprehensive suite of tools to study several properties of interest for the Heisenberg model, including energy, magnetization, specific heat capacity, and magnetic susceptibility. For the Metropolis algorithm, we allow for different sampling procedures, including a 180° and fully random spin-flip as well as small step and Gaussian distributed trial moves. Additionally, we also implement an Adaptive Metropolis algorithm that continuously optimizes the spin-flip acceptance rate. Our model includes variable interaction strength, an external magnetic field, and global magnetic anisotropies, and supports varying cubic lattice sizes and different dimensions ($D = 1, 2, 3$) with individual lattice lengths (L_i). We study both the ferromagnetic–paramagnetic phase transition and related critical phenomena, as well as the dynamical properties of de-magnetization for different internal anisotropies and temperatures. All calculations were performed in parallel on the Euler cluster.

Contents

1. Introduction

Lattice models – discrete geometrical structures which originated as a description of atoms in solid state physics – are one of the most important simplifications in many areas of physics. These models are used to understand the formation of order and disorder in crystals, to predict phase transitions of magnetic materials, or to model interactions of particles. In this project, we are investigating the three-dimensional **Heisenberg model**, a lattice model from statistical physics used to model ferromagnetism. In the first chapter (Background), we provide an introduction into the Heisenberg model. Additionally, we explain the phenomenon of phase transitions determined by their order parameter, and introduce critical exponents for physical observables of interest such as the magnetization, susceptibility, and specific heat capacity. Then, we present the Metropolis and Wolff algorithms that we use to simulate thermal processes in the Heisenberg model. Next, we introduce our Implementation and explain how we compare the different algorithms. The section on Result concerns convergence, thermalization, and Monte Carlo phase space sampling, in addition to critical behaviour at the phase transition. We show that it is possible to study systems of different dimensionality, and include an external magnetic field and magnetic anisotropies into the lattice. This enables us to study spin models on different lattices with richer physics in two-dimensional systems or to investigate the competition of order and disorder with magnetic fields and internal anisotropies. In our Summary and Outlook, we resume our findings and provide ideas for future investigations.

2. Background

i. Heisenberg Model

The most basic ingredient of our model is a **lattice** Λ. It is defined as a periodic arrangement of sites labeled by their respective coordinates. We only consider cubic lattices of side lengths $L_x \times L_y \times L_z$ with sites

$$(i,j,k) \in [1,...,L_x] \times [1,...,L_y] \times [1,...,L_z]\,. \quad (1)$$

Each lattice site is occupied by a **spin** $\vec{\sigma}$. In contrast to quantum mechanics, we are not dealing with operators which act on states. Instead, we consider classical spins: $\vec{\sigma} = (\sigma_x, \sigma_y, \sigma_z)$, where $\sigma_i \in \mathbb{R}$. This means that each spin is a vector, pointing in a certain direction, and a superposition of states is not allowed. Additionally, we require the spins to be normalized: $|\vec{\sigma}| = 1$. In principle, this means that we allow each spin to rotate freely on the unit sphere. However, we also have to model the interactions between different spins. Depending on the energetic cost of being aligned in a certain direction, the spins will favor certain configurations. The Hamiltonian H[1] determines the **energy** E of the lattice Λ for a given spin configuration $\vec{\boldsymbol{\sigma}}$:

$$H_{\text{Heisenberg}}(\vec{\boldsymbol{\sigma}}) = -J \sum_{\langle i,j \rangle} \vec{\sigma}_i \cdot \vec{\sigma}_j - \vec{h} \cdot \sum_i \vec{\sigma}_i \quad (2)$$

where the scalar product $\vec{\sigma}_i \cdot \vec{\sigma}_j$ between two vectors describes how much those spins overlap. Here, we only consider the interaction of two nearest neighbor spins $\langle i,j \rangle$ with a certain interaction strength J as well as the interaction of each spin $\vec{\sigma}_i$ with an external magnetic field $\vec{h}$. In eq. 2, we have to sum over all nearest neighbors, or equivalently, over all bonds between lattice sites. For a 3d lattice of lengths L_x, L_y, L_z, this means that the sums in eq. 2 can be computed as following:

$$\sum_i \vec{\sigma}_i = \sum_{i=1}^{L_x} \sum_{j=1}^{L_y} \sum_{k=1}^{L_z} \vec{\sigma}_{i,j,k} \quad (3)$$

$$\sum_{\langle i,j \rangle} \vec{\sigma}_i \cdot \vec{\sigma}_j = \underbrace{\sum_{i=1}^{L_x-1} \sum_{j=1}^{L_y} \sum_{k=1}^{L_z} \vec{\sigma}_{i,j,k} \cdot \vec{\sigma}_{i+1,j,k}}_{\text{all bonds parallel to x}} + \text{etc.} \quad (4)$$

where (i,j,k) are the Cartesian coordinates of each spin in x-, y-, and z-direction, respectively. In eq. 4, we describe the summation for open boundary conditions. For periodic boundary conditions, we have to change $\sum_{i=1}^{L_x-1} \to \sum_{i=1}^{L_x}$, and similar for the summations over y and z. By calculating eq. 2 using the summations in eq. 3 and 4, we can efficiently compute the energy of the Heisenberg model defined on a cubic lattice.

One important quality of the Heisenberg model is that it can be used to study **phase transitions**. In the Ferromagnetic Heisenberg model ($J > 0$), the ground state, i. e. the state of lowest energy, is a spin configuration where all spins point in the same direction $\vec{\sigma}$:

$$\vec{\boldsymbol{\sigma}}_{\text{GS}} = \{\vec{\sigma}_i = \vec{\sigma}\, \forall\, i \in \Lambda\}\,. \quad (5)$$

For low temperatures, we expect to find the system in a state with most of the spins parallel to each other, while thermal fluctuations dominate at high temperatures, leading to a random orientation of the spins. Thus, we predict two **phases**: a ferromagnetic phase and a paramagnetic phase, depending on the temperature. To classify those phases, we introduce the **magnetization**:

$$\vec{M} = \frac{1}{N} \sum_i \vec{\sigma}_i\,, \quad (6)$$

where $\vec{M}$ is the average spin orientation over all lattice sites $N = L_x \cdot L_y \cdot L_z$. In the Ferromagnetic Heisenberg model, $\vec{M}$ can be used as an **order parameter** to distinguish between a low-temperature ordered phase with non-zero magnetization (ferromagnetic phase) and a high-temperature disordered phase with zero magnetization (paramagnetic phase). We denote the **critical temperature** at which the phase transition occurs as T_C. One may be tempted to ask what the spin direction $\vec{\sigma}$ in the ferromagnetic ground state is. Indeed, there does not exist a preferred direction. This naturally leads to the continuous $O(3)$-symmetry for the 3d Heisenberg model without external field $\vec{h} = 0$. If we rotate all spins in our lattice by the same angles, the total energy does not change. However, if $\vec{h} \neq 0$, this symmetry is externally broken. In general, we expect $M \to 1$ for $T \to 0$ and $M \to 0$ for $T \to \infty$ where $M = \left\|\vec{M}\right\|$. For a quantum mechanical lattice model with spin operators whose components do not commute, this symmetry leads to the phenomenon of collective excitations described by quasiparticles. For example, a quantized spin density wave can be described by *magnons*. Similar excitations can also be studied in the classical Heisenberg model. This is often used to model the semi-classical limit of corresponding quantum systems.[1] As such, classical models are used to simulate features such as frustration[2,3] or perform calculations on interesting systems such as different lattices.[4,5]

[1] H is a function which returns a scalar E, defined to be the energy, given a configuration of spins $\boldsymbol{\sigma} = \{\sigma_1, ..., \sigma_N\}$ for a lattice of N spins.

ii. Monte Carlo Phase Space Sampling

In order to perform calculations with the Heisenberg model, one needs to compute expectation values of quantities such as energy or magnetization. In general, the **expectation value** of an observable X is defined as follows:

$$\langle X \rangle = \sum_i p_i X_i \,. \tag{7}$$

Here, p_i is the probability that the observable X has the value X_i. The sum $\sum_i$ runs over all possible values of X, where we have to ensure that the probabilities are normalized: $\sum_i p_i = 1$. In statistical physics, we compute expectation values by means of the **partition sum** Z:

$$Z = \sum_i e^{-\beta E_i} \,, \tag{8}$$

where the Boltzmann factor $e^{-\beta E_i}$ contains the thermodynamic beta or inverse temperature $\beta = \frac{1}{k_\mathrm{B} T}$ and the energy E_i of a certain microstate i, i.e., a certain spin configuration $\boldsymbol{\sigma}_i$. To calculate the probability of a macrostate, e.g., a state of certain energy (which can be achieved by different spin configurations), we have:

$$p_i = \frac{1}{Z} e^{-\beta E_i} \,. \tag{9}$$

The expectation values of observables such as energy can be computed if we know Z. For the energy, we have

$$\langle E \rangle = -\frac{\partial \log Z}{\partial \beta} \quad \text{with} \quad Z = Z(\beta) \,. \tag{10}$$

While it is generally possible to compute Z exactly for small systems with finite degrees of freedom,[2] the complexity of this problem grows exponentially with the lattice dimensions. Moreover, we have to consider that each spin has infinitely many possible states in the Heisenberg model.[6] Again, we can only compute Z exactly for small systems. This explains why we need numerical algorithms to simulate this lattice. Often, we are relying on Monte Carlo methods to sample the phase space, i.e., approximate the exact expectation values of observables. The success of random Monte Carlo sampling stems from the fact that we are usually only exploring a small part of the total phase space.

[2] If we consider the Ising model, where $\sigma_i = \pm 1$ for every site instead of $\vec{\sigma}_i$, we know that Z is a sum over 2^N terms, with N being the total number of spins. Thus, we could generally compute the expectation values of small Ising models exactly. However, even for a small cubic lattice of side length $L = 5$ in three dimensions, we already have $2^{5 \cdot 5 \cdot 5} \approx \mathcal{O}(10^{37})$ different possible configurations, which is impossible to handle. This explains why for almost all lattice systems, we need another way to calculate expectation values.

For sufficiently large N, we can estimate the expectation values of observables X such as energy E and magnetization M[3] as follows:

$$\langle X \rangle = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{11}$$

and

$$\left\langle X^2 \right\rangle = \frac{1}{N} \sum_{i=1}^{N} X_i^2 \tag{12}$$

under the assumptions that:

(a) $p_i = \frac{1}{N}$, the probability of each measurement is equal,

(b) X_1 is measured for a system in thermal equilibrium,

(c) $\langle X_i X_{i+1} \rangle \approx 0$, i.e., there exist no temporal correlations.

In order to fulfill (a), we need to ensure that the measurements are uncorrelated, which is equivalent to (c). We can use the **non-linear correlation function** to study the deviation of an observable $X(t)$ from its steady state value $X(t \to \infty)$ [7]

$$\phi_{X,\mathrm{nl}}(t) = \frac{\langle X(t) \rangle - \langle X(t \to \infty) \rangle}{\langle X(t_0) \rangle - \langle X(t \to \infty) \rangle} \,, \tag{13}$$

where t_0 is the initial time and $t \to \infty$ is the time when we have reached thermal equilibrium. We can see that

$$\phi_{X,\mathrm{nl}}(t_0) = 1 \quad \text{and} \quad \phi_{X,\mathrm{nl}}(t \to \infty) \to 0 \,. \tag{14}$$

One idea to determine thermal equilibrium is to measure the energetic variance (which is proportional to the specific heat capacity c_V, quantity that is constant in thermal equilibrium), depending on the number of steps after starting from a specific lattice configuration, and set an error on σ_{c_V}. Then, one could argue that thermal equilibrium is reached when the system's specific heat capacity has a low-enough variance. However, we can also try to determine the non-linear correlations directly. We expect that:

$$\phi_{X,\mathrm{nl}}(t) \sim e^{-t/t_0^{\mathrm{nl}}} \tag{15}$$

where t_0^{nl} is the **non-linear correlation time**, which is also defined as: [7]

$$t_0^{\mathrm{nl}} = \int_0^\infty \mathrm{d}t' \, \phi_{X,\mathrm{nl}}(t') \,, \tag{16}$$

because we know that in general:

$$\int_0^\infty \mathrm{d}t \, e^{-t/\tau} = \tau \,. \tag{17}$$

[3] For the Heisenberg model, we have three magnetization components M_i where $\langle X \rangle$ is a vector and $\left\langle X^2 \right\rangle$ a scalar.

We can approximate τ (and hence t_0^{nl}) and with N_{cutoff} discrete values ϕ_i:[8]

$$\tau \approx \frac{1}{2} + \sum_{i=1}^{N_{\text{cutoff}}} \phi_i \,. \tag{18}$$

As the correlations increase close to the critical temperature, we expect t_0^{nl} to increase when approaching T_C. This is know as the **critical slowing down** of the dynamics:[9]

$$t_0^{\text{nl}} \sim \lim_{T \to T_C} |T - T_C|^{-z} \,, \tag{19}$$

where z is the non-linear dynamical critical exponent.[7] Thus, is becomes increasingly hard to calculate expectation values upon approaching T_C. This power law dependence is due to a critical behaviour general to the Heisenberg model. Moreover, we expect the correlation time to depend on the lattice dimensions, because a larger lattice leads to more steps needed to reach thermal equilibrium. Hence, we have to determine t_V for a given volume $V = L^3$ and extrapolate it for a different volume $V' = L_x \cdot L_y \cdot L_z$:

$$t_{V'} = \frac{V'}{V} t_V \,. \tag{20}$$

Similar to the previous discussion, we also have to ensure that lattice configurations are uncorrelated when we measure a series of observables to calculate expectation values. The **linear correlation function** is defined as:

$$\phi_{X,\text{l}}(t) = \frac{\langle X(t_0) \cdot X(t) \rangle - \langle X(t_0) \rangle^2}{\langle X(t_0)^2 \rangle - \langle X(t_0) \rangle^2} \,, \tag{21}$$

where t_0 is the time needed to reach thermalization. Similarly to $\phi_{X,\text{nl}}(t)$, we again expect:

$$\phi_{X,\text{l}}(t) \sim e^{-t/t_0^{\text{l}}} \,. \tag{22}$$

Thus, we can determine the **linear correlation time** t_0^{l} with the same approximation, see eq. 18. Given a series of measurements $X_1, \ldots, X_N$ which can be used to compute expectation values of the respective observables, see eq. 11 and 12, we are also interested in the variance:

$$\sigma_X^2 = \left\langle X^2 \right\rangle - \langle X \rangle^2 \,. \tag{23}$$

The **fluctuation-dissipation theorem** provides a useful tool to use such fluctuations in energy or magnetization and associate them with other quantities such as the specific heat or susceptibility. Thus, we can relate variances in energy E for a given temperature T to the specific heat capacity c_V:

$$c_V = -\frac{1}{k_{\text{B}} T^2} \left(\left\langle E^2 \right\rangle - \langle E \rangle^2 \right) . \tag{24}$$

Often, one normalizes c_V by the total number of spins $N = L_x \cdot L_y \cdot L_z$. Similarly, we can also measure the magnetic susceptibility χ by calculating variances in the magnetization M:

$$\chi = \frac{1}{k_{\text{B}} T} \left(\left\langle M^2 \right\rangle - \langle M \rangle^2 \right) . \tag{25}$$

Note that the magnetic susceptibility is defined even for $h = 0$ (no external field).

iii. Phase Transitions and Critical Exponents

The phase transition from a ferromagnetic to a paramagnetic phase can be measured by interpreting the magnetization M as an order parameter. M goes to zero at the critical temperature T_C (or becomes significantly smaller in finite-size systems). If there is a discontinuity (e.g., jump) in the order parameter at T_C, then we have a first-order (discontinuous) phase transition. Otherwise, we are observing a second-order (continuous) phase transition, which is the case for the Heisenberg model, where $M(T)$ is a continuous function. Moreover, the specific heat capacity diverges (or becomes maximal), but there is no jump, in contrast to first order transitions. This criterion (i.e., observing the maximum of $c_V(T)$) can be used to determine the critical temperature T_C. The divergent behaviour of quantities such as the specific heat, susceptibility, or correlation length when approaching T_C can be modelled by power-laws with so-called **critical exponents**.[10] For example, we are interested in the critical exponent of the order parameter $\beta \in \mathbb{R}$:

$$M(t) \sim \lim_{t \to 0} |t|^{-\beta} \quad \text{with} \quad t := \frac{T - T_C}{T_C} \,, \tag{26}$$

which is only defined for $t < 0$ as $M(t > 0) = 0$. Similarly, we would like to determine the function

$$c_V(t) \sim \lim_{t \to 0} |t|^{-\alpha} \,, \tag{27}$$

where α is the critical exponent for the specific heat. We write α for $T > T_C$ and α' for $T < T_C$. The critical exponents γ and γ' for the magnetic susceptibility

$$\chi(t) \sim \lim_{t \to 0} |t|^{-\gamma} \tag{28}$$

are defined analogously.[11] In general, spins aggregate to clusters with exponentially decaying correlation lengths ξ. By introducing ν, ν' for the correlation length:[11]

$$\xi(t) \sim \lim_{t \to 0} |t|^{-\nu} \,, \tag{29}$$

we can characterize the increase of correlations close to the critical point. For larger systems, we expect a more

pronounced divergence peak.[7] The height of the susceptibility peak scales with $L^{\gamma/\nu}$, whereas the width of the critical region decreases with $L^{-1/\nu}$.[7,12] Hence, we can use the **finite size scaling relation** of the susceptibility to determine the critical exponents γ and ν:

$$\chi \sim L^{\gamma/\nu} f((T-T_C)L^{1/\nu}), \tag{30}$$

where f is the scaling function.[7] Another useful quantity is the **Binder cumulant**:

$$U_L = 1 - \frac{\langle M^4 \rangle}{3\langle M^2 \rangle^2} \tag{31}$$

which is defined with the higher-order moments of the magnetization $\langle M^2 \rangle$ and $\langle M^4 \rangle$ to determine T_C precisely since U_L is independent of L at T_C.[7] Hence, we can plot $U_L(T)$ for different system length L against the temperature and determine the critical temperature T_C as the point where the different $U_L(T)$ curves overlap.[7] Even with periodic boundary conditions, we expect the critical temperature T_C to depend on the dimensionality and system size. As a literature reference, Holm and Janke report $T_C \approx 1.440$ for a lattice size L of order $\mathcal{O}(10)$,[10] which agrees with the more general result of Peczak et al. of $T_C \approx 1.4432$.[13] In general, we expect the critical temperature to be larger for smaller lattice sizes.[10,14]

iv. Extensions of the Heisenberg Model

Naturally, one may try to extend the (classical) Heisenberg model to describe lattice systems with additional constraints and effects. One idea is to include a **magnetic anisotropy**, i.e., an additional term in the Hamiltonian such each spin's magnetic moment has a preferred direction.[15] In contrast to an isotropic model, this anisotropy leads to differences in the susceptibility depending on the rotation with respect to the external field. The easy axis is the direction along which a sample can be magnetized the easiest. We use $\vec{k}$ as the vector of the magnetic anisotropy and define the Hamiltonian of the **anistropic Heisenberg model** as follows:

$$H_{\text{Heisenberg}} = -J \sum_{\langle i,j \rangle} \vec{\sigma}_i \cdot \vec{\sigma}_j - \vec{h} \cdot \sum_i \vec{\sigma}_i - \sum_i (\vec{k} \cdot \vec{\sigma}_i)^2 . \tag{32}$$

Again, we can evaluate eq. 32 efficiently by performing an additional summation over all lattice sites $\sum_i$, analogous to eq. 2. To include **local magnetic impurities**, we can let $\vec{k}$ be position dependent:

$$\vec{k} \to \vec{k}_{i,j,k} = \begin{cases} \vec{k}, & \text{if } (i,j,k) \in \Lambda_{\text{impurity}}, \\ \vec{0}, & \text{otherwise}. \end{cases} \tag{33}$$

Here, $\Lambda_{\text{impurity}}$ is a region where the magnetic behaviour differs from the rest of the lattice Λ. Similarly, we can also include anisotropies in the interaction strength $J \to J_{i,j,k}$ and magnetic field $\vec{h} \to \vec{h}_{i,j,k}$.

v. Monte Carlo Algorithms

In the previous section, we describe why it is impossible to study systems that are not small without making use of numerical algorithms. Such algorithms create states in thermal equilibrium that follow a Boltzmann distribution. In our project, we implement two well-established algorithms and extend them to include additional effects such as external magnetic fields and magnetic anisotropies.

v.1. Metropolis Algorithm

The **Metropolis Algorithm** was first published in 1953 to simulate statistical equations of state.[16] In the following pseudocode (Algorithm 1), we describe the Metropolis algorithm for a given number of maximal steps N_{max}.

Algorithm 1 Metropolis Algorithm

Require: Initial lattice configuration
  N = 1
  while $N < N_{\text{max}}$ **do**
    Choose random spin at site i
    Compute ΔE upon flipping spin: $\vec{\sigma}_i \to -\vec{\sigma}_i$
    Accept spin flip with probability:
      $p = \min(1, e^{-\beta \Delta E})$
    Continue with (new) lattice configuration
    N += 1
  end while

For each run N, we select a spin at random and calculate the change in energy (determined by the system's Hamiltonian) upon flipping this spin. If energy is gained ($\Delta E > 0$), this spin flip is always accepted, otherwise we will only continue with this new lattice configuration with a probability $p = e^{-\beta \Delta E}$. Here, we can use different types of spin flips. The most basic type of spin flip is a **perfect spin-flip** by 180°:

$$\vec{\sigma}_i \to -\vec{\sigma}_i . \tag{34}$$

Obviously, this algorithm is not guaranteed to converge to thermal equilibrium as we can only calculate 2^V different spin configurations. One generalization is the random spin-flip move:

$$\vec{\sigma}_i \to \vec{\sigma}_i' , \tag{35}$$

where $\vec{\sigma}_i'$ is randomly chosen. While this algorithm might perform well in the disordered phase, it will certainly be

inefficient for low-temperature regions where spins are preferably aligned parallel to each other, especially if all spins point in the same direction in our initial lattice. In order to achieve satisfactory converge behaviour, Haario et al. proposed an **Adaptive Metropolis Algorithm** [17] which is based on the Random Walk Metropolis algorithm, [16] see algorithm 1. Here, the *trial move* is often chosen to be a **Gaussian move** instead of a perfect spin flip. This means that the new spin direction $\vec{\sigma}'$ after a flip is given by:

$$\vec{\sigma}' = \frac{\vec{\sigma} + \alpha\vec{\Gamma}}{\left|\vec{\sigma} + \alpha\vec{\Gamma}\right|}, \tag{36}$$

where $\vec{\sigma}$ is the previous spin direction, $\vec{\Gamma}$ is a random vector whose components are drawn from a Gaussian distribution, and α is a variable which is proportional to the width of the cone around the initial spin direction. Thus, we can adjust the acceptance rate by changing α. In every step, we change $\alpha \to \alpha'$ such that:

$$\alpha' = \alpha \cdot f\,, \quad f = \frac{1}{2\cdot(1-R)}\,, \tag{37}$$

where R is the acceptance rate in the previous step with given α. We determine R_s (where $n \in \mathbb{N}$ is the current step) as following:

$$R_{n\geq 2} = \frac{\sum_{i=1}^{n-1} R_i}{n-1} \text{ with } R_i = \begin{cases} 1, \text{ if accepted}\,, \\ 0, \text{ if declined}\,, \end{cases} \tag{38}$$

where each R_i is one (zero) if the spin flip is accepted (declined). We determine $R_1 \in \{0,1\}$ for the first step and process according to eq. 38. With this procedure, we can fulfill the **Golden Rule** of the Metropolis algorithm, stating that an acceptance rate of 50% leads to maximal efficiency. [18,19] This algorithm has shown to be more efficient than other commonly used spin update algorithms, independently of temperature and anisotropy, leading to lower correlation times and a faster convergence towards thermal equilibrium. [15] In the following, we will always refer to the *Metropolis* and *Adaptive Metropolis* algorithms when the trial moves are a small-step move with a randomly sampled opening angle of $\theta' = 10°$ around the old spin and a Gaussian move (see eq.36), respectively.

v.2. Wolff Algorithm

The main disadvantage of the Metropolis algorithm is critical slowing down near a phase transition. The **Wolff Algorithm** [9] updates clusters of sites instead of single spins to overcome this challenge. Long autocorrelation times can be avoided as the converge only shows a weak dependence on the lattice size. However, using the Wolff algorithm is discouraged far away from critical points, or for small systems, with autocorrelation times that are negligible for the convergence behaviour. [7] In the following pseudocode (Algorithm 2), we explain the Wolff algorithm which we implemented by means of a stack.

Algorithm 2 Wolff Algorithm

Require: Initial lattice configuration
  N = 1
  Stack = {}
  while $N < N_{\text{max}}$ **do**
    Choose random spin σ_r and random lattice site i
    Add i to the stack
    Flip σ_i w.r.t. the hyperplane orthogonal to r:
      $\sigma_i \to \sigma_i - 2(\sigma_i \cdot r)r$
    while Stack is not empty **do**
      Pop site j from the stack
      if Site j is not marked **then**
        Mark site j
        Check every neighbor k of j
        Activate bond $k-j$ with probability:
        $p_r(\sigma_j, \sigma_k) = 1 - e^{\min(0, 2\beta(r\cdot\sigma_j)(r\cdot\sigma_k))}$
        if Bond is activated **then**
          Add neighbor k to stack
          Mark site k
        end if
      end if
    end while
  end while

In every algorithmic step, we choose a random spin σ_r and a random lattice site i, flip the spin σ_i at site i with respect to σ_r, and check the bonds to all neighbors at sites k. The stack is used to keep track of all remaining neighbors until all bonds on the surface of our three-dimensional cluster have been checked. While in the Metropolis algorithms, spins are not always flipped, depending on the energy difference, this is not the case for the Wolff algorithm. Instead, we are always flipping a cluster of spins, where the cluster size is related to the energy difference between spins.

3. Implementation

i. Code Structure

Our simulation of the Heisenberg model was implemented entirely in C++. The **lattice** is implemented with a class `lattice`. This class is defined by:

- L_x, L_y, L_z: the dimensions of the lattice,
- `bc` = `o,p`: the boundary conditions (open and periodic).

We have implemented the boundary conditions by always summing over L_x many bonds in the x-direction, even if we select open boundaries, and choosing the spin $\vec{\sigma}_{L_x+1}$ at site L_x+1 (which is calculated in the sum term $\vec{\sigma}_i\vec{\sigma}_{i+1}\forall i \in [1, L_x]$) to be $\vec{\sigma}_1$ for periodic boundary conditions or $\vec{0}$ for open boundary conditions, respectively. The following **algorithm** functions:

- `void Metropolis(&lattice, T, time, steps, J, h)`
- `void Metropolis_Adaptive(&lattice, T, time, steps, J, h)`
- `flt Wolff(&lattice, T, time, steps)`

are called with a reference to the lattice (`Lattice &lattice`), the current temperature (`flt T`), an amount of time (`flt time`) which they should run for, and the number of steps (`int steps`). The algorithms are implemented such that they stop when they run out of `time` or have carried out all `steps`. Thus, we set `time` = ∞ for a certain no. of `steps` = $n_{\texttt{steps}}$, and vice versa. It is important to note that a step in the Metropolis algorithm is not comparable to the Wolff algorithm. Similarly, the algorithms perform a different number of bond updates during the same amount of time. All algorithms also require the interaction strength `J`, the current magnetic field vector (`h`, with three different components h_x, h_y, h_z), and the anisotropy vector (`k`, again with three components k_x, k_y, k_z) to calculate the energetic cost of a spin flip. When calling the respective function, the algorithm changes part of the spin configuration through its reference to the lattice (`&lattice`).[4] In addition, the Wolff algorithm also returns the susceptibility. We always used the Mersenne Twister pseudo-random generator (`mt:19937`) to get random numbers.

[4] Alternatively, we could also return an array which represents the lattice in each step. However, this is inefficient with respect to memory as many copies of the lattice are created.

ii. Thermalization

One goal of this project is to determine when our spin configurations approach thermal equilibrium depending on the respective algorithm. For a qualitative analysis, we calculate the variance of a specific observable such as the specific heat c_V which becomes constant in thermal equilibrium and set a cutoff value for its variance σ_{c_V}. With the function:

- `loop_algorithm(algorithm, quantity,T,`N_s`,`$N_{\max}$`)` ,

we can perform a Markov-chain Monte Carlo simulation of a certain algorithm (`algorithm`). We provide a lattice with given interaction strength and field, the maximal number of steps N_{max} that the algorithm should run for, and the step size N_s after which the current values of a quantity $X = M, E$ should be saved in another array. This means that we measure the respective quantity `no_of_steps` = `round(`$N_s/N_{\max}$`)` many times. For example, if we provide a `quantity`= M, N_s = 10 and $N_{\max}$ = 30, the algorithm creates an `array_of_steps` = $[10, 20, 30]$ where the magnetization M is measured after 10, 20 and 30 steps. Then, `loop_algorithm(...)` returns an array `return_array` = $[M_{\text{steps=10}}, M_{\text{steps=20}}, M_{\text{steps=30}}]$ with values which can be plotted against `array_of_steps`. In a second step, we can loop through this array of measurements after a certain time or number of steps and provide arrays of weighted mean values and weighted standard deviations.[5] For example, for a given `array_of_steps` = $[10, 20, 30]$ and the `values_array` = $[M_{\text{steps=10}}, M_{\text{steps=20}}, M_{\text{steps=30}}]$, we are interested in `average_array` = $[\overline{M}_{\text{steps=10}}, \overline{M}_{\text{steps=20}}, \overline{M}_{\text{steps=30}}]$ where $\overline{M}_{\text{steps=10}} = M_{\text{steps=10}}$, $\overline{M}_{\text{steps=20}} = {}^{1\cdot M_{\text{steps=10}}+2\cdot M_{\text{steps=20}}}/_{3}$, etc. For a given set of data points $x_i \in [x_1, ..., x_n]$, the mean value $\overline{x}$ is defined as:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \,. \tag{39}$$

Additionally, we have to include weights to correct for the fact that the measurement after 20 steps already includes

[5] We can only use unweighted averages if the measured observable values (for example magnetization values) are uncorrelated. However, this is not the case when we determine the convergence behaviour. If we would not use weighted averages, we would consider the first magnetization value (for example after one algorithmic step) with similar importance as the last one (which might be in thermal equilibrium). Obviously, we need to correct this, for example by weighing the measurements with the number of algorithmic steps.

the observable value at 10 steps. Thus, we calculate:

$$\overline{x}^{\mathrm{w}} = \frac{1}{n \cdot N_{\mathrm{sum}}} \sum_{i=1}^{n} i \cdot N_s \cdot x_i \,, \tag{40}$$

where N_{sum} is the sum of all weights, and $i \cdot N_s$ is the total number of steps that have been performed when the i-th step was measured. Similarly, if we provide the `array_of_steps`, `values_array`, and the previously calculated `array_of_means = average_array(...)`, we would like to know `std_array` = $[\sigma_{M,\mathtt{steps=10}}, \sigma_{M,\mathtt{steps=20}}, \sigma_{M,\mathtt{steps=30}}]$. The standard deviation of an array of data points $x_i \in [x_1, ..., x_n]$ with mean value $\overline{x}$ is calculated for the n-th sample as following:

$$\sigma_{X,n} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - \overline{x})^2} \,, \tag{41}$$

with $\sigma_{X,1} = 0$. We are using the biased sample variance $\sigma^2_{X,n}$ rather than the unbiased samples variance (where the normalization is given by $\frac{1}{n-1}$ instead of $\frac{1}{n}$) because our samples $[x_1, ..., x_n]$ are not drawn independently, but instead form a Markov chain. Hence, we also have to consider that our samples are weighted by the total number of steps $i \cdot N_s$ that have been performed for the i-th step. We can calculate the standard deviation with weights given by the array $[N_s, 2 \cdot N_s, ..., N_{\mathrm{max}}]$ as follows:

$$\sigma^{\mathrm{w}}_{X,n} = \sqrt{\frac{1}{n \cdot N_{\mathrm{sum}}} \sum_{i=1}^{n} i \cdot N_s \cdot (x_i - \overline{x}^{\mathrm{w}})^2} \,. \tag{42}$$

In Fig. 1, we sketch the procedure to determine the thermalization after N_{eq} steps and the subsequent run of a certain algorithm to determine expectation values $\langle X \rangle$ of a given quantity. First, we measure after which number of steps N the variance of a certain observable ($X = c_V, \chi$) falls below an error (corresponding to a certain $\sigma_{c_V}, \sigma_\chi$) that we have set previously. Next, with an order of magnitude for N in mind, we can calculate the number of steps precisely by means of the non-linear correlation time, see eq. 13. Obviously, we have to check that the mean value $\overline{X}$ converges to the anticipated equilibrium value $X_{\mathtt{eq}}$. With this procedure, we determine a fixed number of steps $N_{\mathtt{eq}}$ after which the algorithm has reached thermal equilibrium. We expect $N_{\mathtt{eq}}$ to be dependent on the specific algorithm used. In contrast to the Metropolis algorithm, each step in the Wolff algorithm generally consists of a different amount of operations. Hence, a number of steps N of each algorithm is preferable over a fixed amount of time. The thermalization condition has to be checked

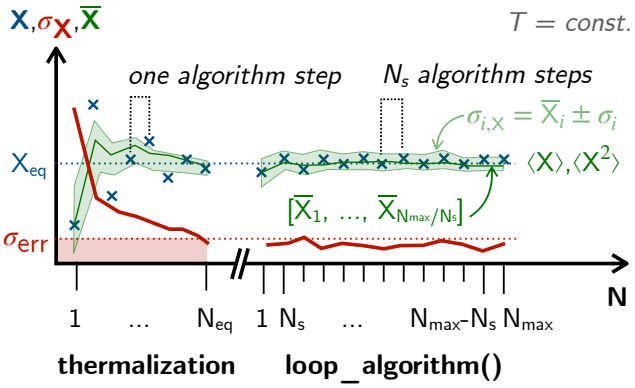

Figure 1: *Measuring the observables: First, we wait for the thermalization to set in after a certain number of steps $N_{\mathtt{eq}}$. This is confirmed by plotting a quantity $X = c_V, \chi$ which converges upon reaching thermal equilibrium. Then, the variance of X goes to zero. After reaching thermal equilibrium, we measure an observable of interest $X = M, E$ (blue cross) every time after running the algorithm for N_s steps and repeat this until we have performed $N_{\mathtt{max}}$ steps. The resulting array of values $[\overline{X}_1, ..., \overline{X}_{N_{\mathtt{max}}/N_s}]$ is used to determine the expectation values $\langle X \rangle$ (green line) and $\langle X^2 \rangle$ as well as the variance (light-green area) up to the i-th step. After $N_{\mathtt{max}}$ steps, we can determine the order parameter M and energy E as well as specific heat capacity c_V and magnetic susceptibility χ.*

for different temperatures as the algorithms converge differently depending on T. Also, we have to consider that N_{eq} will scale linearly with the system size, i.e., volume. To reduce the number of steps $N_{\mathtt{eq}}$, we will use the following **initial lattice configurations**:

- $T < T_C$: all spins along the z-direction,
- $T > T_C$: all spins randomly orientated.

Since we will never reach T_C exactly, there is no need to include the case $T = T_C$.

iii. Calculating the Observables

In Fig. 1, we explain why the procedure for approximating expectation values in the thermalized system (after $N_{\mathtt{eq}}$ steps) is equivalent to the way how we determine the thermalization in the first place, apart from the specific quantity that we measure. We start by determining the number of steps $N_{\mathtt{s}}$ between two measurements after which the respective lattices do not show any correlations due to the algorithms. If we choose to measure this quantity

as time (t_s), then we have to ensure that $t_s > \tau_0$ where $\tau_0 = \tau_0^{\alpha}(T)$ is the **autocorrelation time**, which will also depend on temperature, the lattice volume, and the respective algorithm. Moreover, it is important to ensure that the variance for the i-th data point $\sigma_{i,X}$ of all values up to a certain average value $\overline{X}_i$ generally scales as $\sigma_N \sim \frac{1}{\sqrt{N}}$. We expect $N_{eq}^{\alpha}(T)$ to become maximal upon approaching T_C, which is known as critical slowing down.[7] Once we have determined N_{eq} and N_S and we have reached thermal equilibrium, we can measure $X = M, E$ every N_S steps until we have performed N_{max} steps. In $[X_1, ..., X_{N_{max}/N_S}]$, we have saved `round(`$N_{max}/N_S$`)` values of the observable X for every different external condition. The average value of this array is $\langle X \rangle$, and the variance it related to σ_X, i.e., the specific heat capacity or susceptibility. We can calculate $\langle X^2 \rangle$ and $\langle X^4 \rangle$ analogously. When measuring the order parameter of the phase transition, it is preferred to use $\langle M \rangle$ and not M_i for a single measurement. Similarly, we are using $\langle E \rangle$ instead of E_i to show how the total energy changes, for example when changing temperature.

4. Results

In the following, we always use units of $J = 1$ and $k_B = 1$, and thus $\beta = \frac{1}{T}$. Unless otherwise noted, $\vec{h} = 0$ and $\vec{k} = 0$. For the initial lattice configuration, we set $\sigma_i = (0,0,1)\,\forall i \in \Lambda$ (i. e. fully magnetized along z) for $T < T_C$ and use random spins at every site for $T > T_C$.

i. Convergence

First of all, we have to determine convergence criteria for the thermalization. In Fig. 2, we present the convergence of the specific heat capacity for different algorithms depending on the number of steps. For a given lattice size $L = 8$ and different temperatures $T = 0.3, 1.44$ and 2.37, we determine the weighted mean energy E after a certain number of algorithmic steps N. The energy variance is proportional to the specific heat capacity c_V, which converges to its equilibrium value after a sufficient amount of steps. We see that for low temperatures, the Metropolis algorithm behaves similar to the Wolff cluster algorithm. In the low-temperature ordered phase, the Metropolis algorithm with small step trial moves leads to faster convergence compared to the Adaptive Metropolis algorithm with a Gaussian move. Close to the critical temperature, we can observe that the Wolff algorithm leads to fast convergence, whereas the Metropolis algorithms suffer from critical slowing down. In the high-temperature disordered phase, the Adaptive Metropolis algorithm shows faster convergence in comparison to the normal Metropolis algorithm, while converging after a similar amount of steps as the Wolff algorithm. Those qualitative results agree with our expectations of the convergence behaviour. In general, it is preferable to use different algorithms depending on the temperature. Our qualitative data (which shows how the variance of E converges) can be compared to the calculated values for the non-linear correlation time (or an amount of steps N_{eq}) after which we consider the system to be thermalized. This gives us a rough estimate of the number of steps that we need to reach thermal equilibrium.

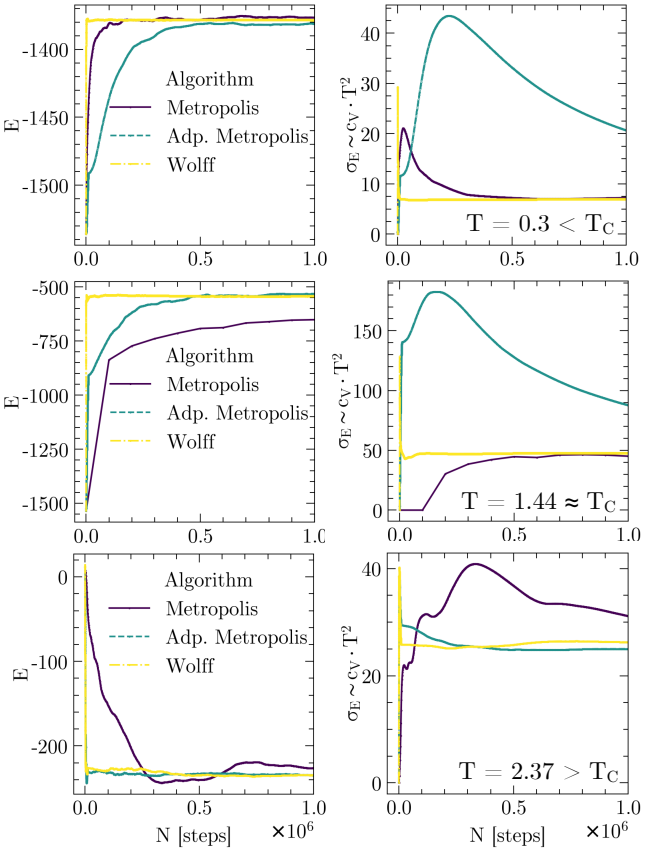

Figure 2: *Energy E and variance $\sigma_E \sim c_v$ (weighted averages) for different algorithms (purple: Metropolis, turquoise: Adaptive Metropolis, yellow: Wolff) after a certain number of algorithm steps N up to $N_{max} = 10^6$ for $T = 0.3 < T_C$ (upper row), $T = 1.44 \approx T_C$ (middle row), and $T = 2.37 > T_C$ (lower row). The convergence behavior of c_V is used to determine thermal equilibrium when $\sigma_{c_V} \to 0$ (not shown). All data is calculated for $L = 8$.*

In Fig. 3, we present an example of a non-linear correlation function $\phi(N)$ for up to $N = 200$ and $N = 10000$ steps for the Wolff and (Adaptive) Metropolis algorithms,

respectively. Additionally, we plot an exponential fit with the non-linear correlation time $\tau = 25$ steps for the Wolff algorithm. Usually, one multiplies this value with 3 to correct for errors. All calculations were performed at $T = 2.33$.

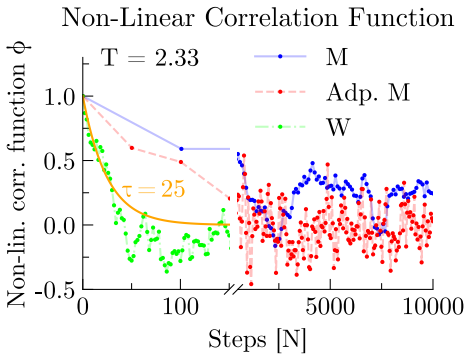

Figure 3: *Non-linear correlation function ϕ as a function of steps N for three different algorithms. Blue: Metropolis (M), Red: Adaptive Metropolis (Adp. M), Green: Wolff (W) for $T = 2.33$, including the exponential decay (orange) for $\tau = 25$ steps and $L = 8$.*

Starting off from a random initial lattice, we see that the Wolff algorithm converges after few steps, whereas the Metropolis and Adaptive Metropolis algorithms take about one to two orders of magnitude more steps to converge. In Fig. 4, we present the non-linear correlation time which is defined as a discrete sum of the non-linear correlation function, see eq. 18.

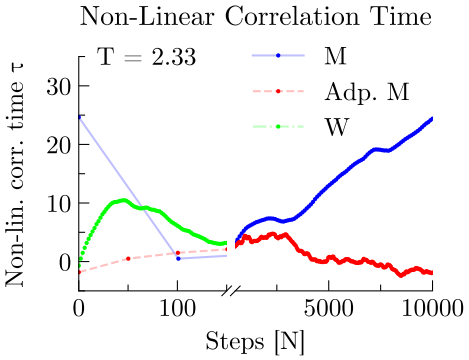

Figure 4: *Non-linear correlation time τ as a function of steps N for three different algorithms. Blue: Metropolis (M), Red: Adaptive Metropolis (Adp. M), Green: Wolff (W) for $T = 2.33$ and $L = 8$.*

With this procedure, we determine the non-linear correlation time N_{eq}, which we present for three different temperatures $T = 0.3, 1.44$, and 2.33 below, at, and above the phase transition, respectively, see Table 1. The calculations were performed for all algorithms on a lattice of size $L = 8$. Those values show the phenomenon of critical slowing down for the Metropolis algorithm: the non-linear correlation increases by a factor of approx. 111. In comparison, we observe that N_{eq} decreases for the Adaptive Metropolis algorithm, and stays approx. constant for the Wolff algorithm.

Algorithm:	**Metropolis**	**Adp. Metropolis**	**Wolff**
$T = 0.3$	$2.7 \cdot 10^4$	$4.5 \cdot 10^4$	$9.6 \cdot 10^1$
$T = 1.44$	$3.0 \cdot 10^6$	$3.0 \cdot 10^4$	$8.0 \cdot 10^1$
$T = 2.33$	$1.5 \cdot 10^5$	$2.0 \cdot 10^3$	$1.2 \cdot 10^3$

Table 1: *Non-linear correlation time: Number of steps N_{eq} needed to reach thermal equilibrium for different temperatures $T = 0.3, 1.44, 2.33$ and all three algorithms. All values were calculated for $L = 8$.*

In Table 2, we present linear correlation times for the same temperatures, which were determined similarly. We see that the amount of steps needed for two lattice configuration to be uncorrelated increases for temperatures close to T_C. Furthermore, we observe that the Wolff algorithm needs approx. three orders of magnitude fewer steps.

Algorithm:	**Metropolis**	**Adp. Metropolis**	**Wolff**
$T = 0.3$	$1.5 \cdot 10^6$	$9.0 \cdot 10^4$	$4.5 \cdot 10^2$
$T = 1.44$	$4.2 \cdot 10^6$	$1.2 \cdot 10^5$	$9.0 \cdot 10^2$
$T = 2.33$	$3.0 \cdot 10^5$	$6.0 \cdot 10^4$	$6.9 \cdot 10^3$

Table 2: *Linear correlation time: Number of steps N_S between two measurements in thermal equilibrium for different temperatures $T = 0.3, 1.44, 2.33$ and all three algorithms. All values were calculated for $L = 8$.*

One important caveat regarding those results is the initial lattice configuration. In order to ensure similar algorithmic performances for $T \gtrapprox T_C$ and $T \lessapprox T_C$, it might be useful to choose a mixture of both initial lattice configurations. For example, one could define:

$$\vec{\sigma}_{i,j,k} = \alpha \vec{\sigma}_z + \beta \vec{\sigma}_{\text{random}} \,, \tag{43}$$

where $\vec{\sigma}_z$ is a spin along z, $\vec{\sigma}_{\text{random}}$ a random (normalized) spin, and $\alpha(T) + \beta(T) = 1$ are choosen such that $\alpha(T \to 0) = 1$, $\beta(T \to \infty) = 1$ and $\alpha(T_C) = \beta(T_C) = 0.5$. Otherwise, the non-linear correlation times of $T \gtrapprox T_C$ and $T \lessapprox T_C$ cannot be compared due to different initial lattices.

Based on the previous calculations, we provide the following estimates for the non-linear and linear correlation

times, N_{eq} and N_{S}, respectively. The following values are based on our results for $L = 8$ and normalized by L^3. We find that the non-linear correlation time per spin at T_C is:

$$\begin{aligned} N_{\text{eq}} \,/\, \text{Spin} &= 5.86{\cdot}10^{3} \quad \text{(Metropolis)}, \\ N_{\text{eq}} \,/\, \text{Spin} &= 5.86{\cdot}10^{1} \quad \text{(Adp. Metropolis)}, \\ N_{\text{eq}} \,/\, \text{Spin} &= 1.56{\cdot}10^{-1} \quad \text{(Wolff)}. \end{aligned} \tag{44}$$

Similarly, we determine the linear correlation time per spin at T_C to be:

$$\begin{aligned} N_{\text{S}} \,/\, \text{Spin} &= 8.20 \cdot 10^{3} \quad \text{(Metropolis)}, \\ N_{\text{S}} \,/\, \text{Spin} &= 2.34 \cdot 10^{2} \quad \text{(Adp. Metropolis)}, \\ N_{\text{S}} \,/\, \text{Spin} &= 1.76 \quad \text{(Wolff)}. \end{aligned} \tag{45}$$

ii. Phase Transition

Perhaps the most interesting feature of the Heisenberg model is its phase transition from a ferromagnetic low-T phase to a paramagnetic high-T phase. In Fig. 5, we present the order parameter M and the magnetic susceptibility χ for a lattice of size $L = 8$ and all three different algorithms. In the left plot, we can clearly recognize a phase transition around $T_C \approx 1.4$, although the divergence of the magnetic susceptibility seems to be closer to $T \approx 1.5$, see also right plot. We have used temperature steps of $\Delta T = 0.01$ around T_C for all calculations.

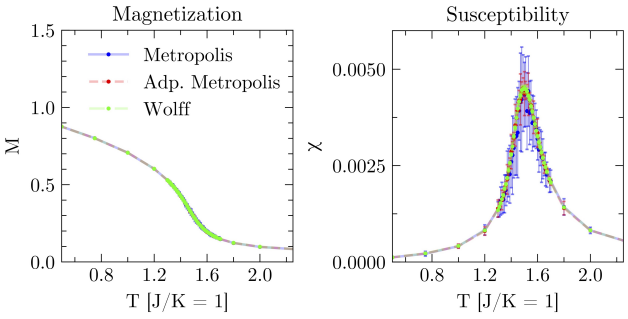

Figure 5: *Magnetization M (left) and magnetic susceptibility χ (right) vs. temperature for different algorithms. All calculations were performed for $L = 8$.*

In Fig. 6, we present the specific heat capacity c_V/N (normalized by the total number of spins N) for different temperatures. We can observe a divergence of c_V (left) and use the maximum value (right) to determine the critical temperature to be $T_C = 1.40 \pm 0.02$. This value is close to the literature, e.g., the study of Holm and Janke [10] that predicts $T_C = 1.4430 \pm 0.0002$ (L of order $\mathcal{O}(10)$), or the result of Peczak et al. [13] with $T_C \approx 1.4432 \pm 0.0002$. Our estimate can be made more precise by performing additional Monte Carlo sampling with smaller ΔT around the critical temperature. The error bars are 1σ standard deviation of the respective quantity M, c_V and χ calculated by performing $M = 36$ parallel runs with the same algorithm and temperature, see also eq. 46 (Appendix). We measured all observables every $N_S = 10^4$ steps and evaluated up to $N_{\text{max}} = 10^8$ steps for the (Adaptive) Metropolis algorithms, leading to 10^4 data points for M and E. For the Wolff algorithm, we used $N_S = 10^2, 10^2, 10^3$ and $N_{\text{max}} = 10^6, 10^6, 10^7$ for low temperatures ($T < 1$), temperatures close to T_C ($1 < T < 2$), and high temperatures ($T > 2$), respectively.

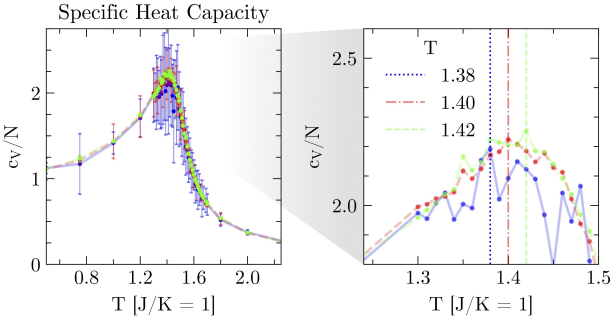

Figure 6: *Left: Divergence of the specific heat capacity per spin c_V/N for different algorithms. Right: Zoomed-in plot without error bars. The critical temperature T_C can be estimated by determining the maximum of $c_V(T)$. All calculations were performed for $L = 8$.*

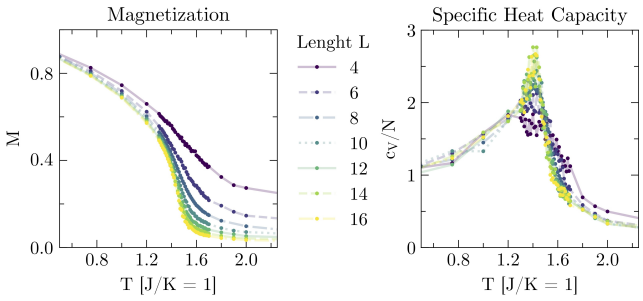

Figure 7: *Simulation for different lattice sizes $L = 4, 6, 8, 10, 12, 14, 16$. Left: Magnetization vs. temperature. Right: Specific heat capacity (normalized by the total number of spins) vs. temperature. All calculations were performed with the Adaptive Metropolis Algorithm.*

After confirming the existence of a phase transition around $T_C \approx 1.4$, we are interested in understanding how the critical phenomena depend on the lattice size L. In Fig. 7, we present the magnetization M and specific heat capac-

ity c_V/N for different lattices from $L = 4$ up to $L = 16$. In the left plot, we can that the magnetization curve $M(T)$ depends on the lattice size. While for smaller lattices, M stays large even in the high-temperature phase, we can observe that $M \to 0$ for $T > T_C$ with increasing lattice size. In the right plot, we can see that the temperature at which c_V diverges seems to be independent of L. Additionally, we see that the specific heat capacity saturates for larger lattices and becomes independent of size, as expected.

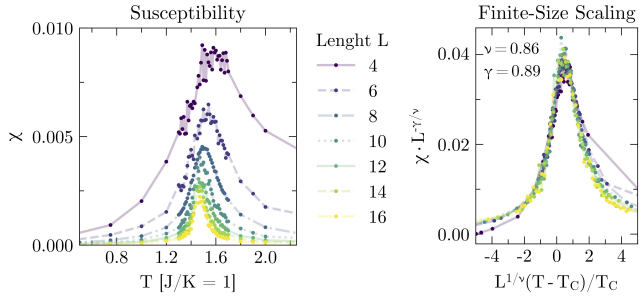

Figure 8: *Left: Susceptibility χ, unnormalized. Right: Finite-size scaling of the susceptibility for critical exponents $\nu = 0.86$ and $\gamma = 0.89$. Simulation with Adaptive Metropolis Algorithm for different lattice sizes.*

In Fig. 8, we present the susceptibility χ for the same lattice sizes. In the left plot, we present χ which shows a characteristic divergence, where the peak temperature seems to converge to T_C for larger lattices. Interestingly, this observable does not collapse when normalized by the total number of spins. Instead, we can use a finite-size scaling relation, see eq. 29. For our calculations, we used $T_C = 1.443$. By systematically changing γ and ν, we find values of $\gamma = 0.89 \pm 0.05$ and $\nu = 0.86 \pm 0.05$ for which $\chi(L, T)$ collapses. The errors are determined based on the linear spacing $\Delta\gamma$ and $\Delta\nu$ of examined (γ, ν)-pairs. However, those critical exponents are in strong disagreement with the literature values of $\gamma = 1.389$ and $\nu = 0.704$. [10] One possible reason is the peak of the susceptibility, which is shifted towards $T_C \approx 1.5$, see also Fig. 5. Indeed, in Fig. 9, we show that the susceptibility peak depends on the lattice size. The maximum susceptibility at $T_{\max}$ deviates slightly from T_C, and there is no clear trend upon changing L. In Fig. 10, we present the scaled susceptibility and show that for different values of T_C, the literature expectations of $\gamma = 1.389$ and $\nu = 0.704$ are still not leading to a data collapse. One possible solution for future investigations is a more extensive range of lengths L, and a larger value of $L_{\min}$.

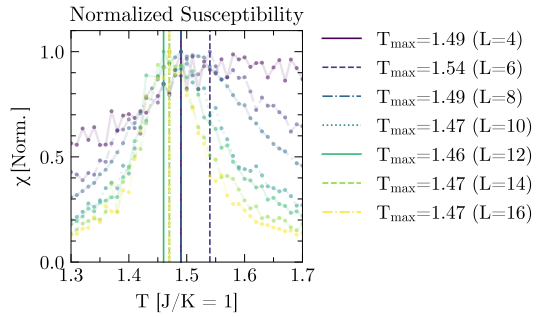

Figure 9: *Susceptibility χ normalized by the maximum value at temperature T_{max}, depending on the lattice size L. Simulation with Adaptive Metropolis Algorithm for different lattice sizes.*

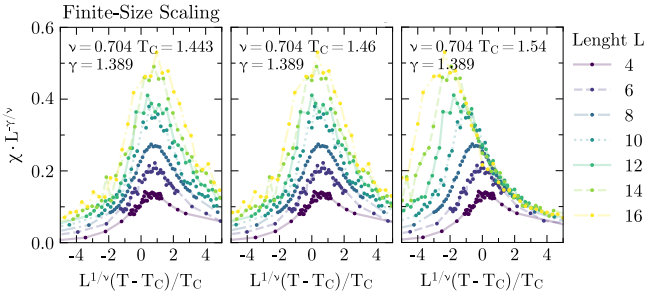

Figure 10: *Finite-size scaling of the susceptibility for critical exponents $\nu = 0.704$ and $\gamma = 1.389$ and $T_C = 1.443$ (left), $T_C = 1.46$ (middle) and $T_C = 1.54$ (right). Simulation with Adaptive Metropolis Algorithm for different lattice sizes.*

Another way to determine ν, the critical exponent of the correlation length, is to apply linear fits to the Binder cumulant $U_L(\beta) \sim m \cdot \beta + b$ for different L. Then, we can determine $1/\nu$ as the slope of $\log m$ vs. $\log L$. With this method, we calculated $\nu = 0.737 \pm 0.076$, which agrees with the literature value of $\nu = 0.704 \pm 0.006$. [10]

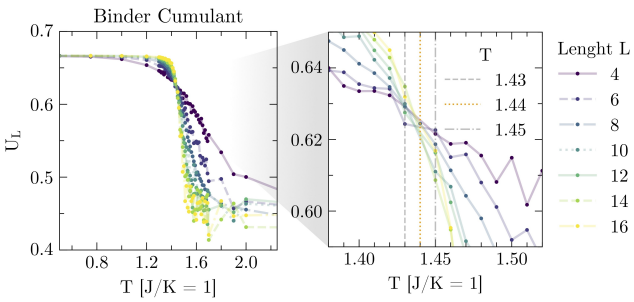

Figure 11: *Simulation with Adaptive Metropolis Algorithm for different lattice sizes. Left: Binder Cumulant U_L vs. temperature. Right: Zoomed-in plot to determine T_C as the temperature where all curves overlap.*

One main interest of Monte Carlo studies is the critical temperature T_C of a phase transition, which can be determined precisely by means of the Binder cumulant, see eq. 31. In Fig. 11, we calculate U_L as a function of temperature for different lattice sizes L. With this method, we can determine the intersection and thus critical temperature to be at $T = 1.44 \pm 0.01$. The error is determined as the distance ΔT to the nearest pair of U_L data points which do not collapse into a single point. This fit matches the literature value of $T_C = 1.4430 \pm 0.0002$.[10]

iii. Critical Exponents

One can describe the behaviour of many observables close to the critical temperature by means of critical exponents. Based on our previous calculations, we can apply a least square fit to determine the critical exponent π for a quantity $X(T)$ where $X = M, c_V, \chi$.[6] In the following Table 3, we list the respective critical exponents (x and x' for $T > T_C$ and $T < T_C$, respectively) with their error Δx based on the least squares fitting procedure. We use the Wolff data from Fig. 5 for our calculations and choose a fitting range close to T_C with $T \in [1.46, 1.6], [1.31, 1.8]$ and $[1.35, 1.65]$ for M, c_V and χ, respectively.

	M	c_V	χ
$\lim_{t\to 0^-}$	/	$\alpha' = -0.827 \pm 0.090$	$\gamma' = 1.120 \pm 0.085$
$\lim_{t\to 0^+}$	$\beta = 0.377 \pm 0.044$	$\alpha = -0.120 \pm 0.065$	$\gamma = 1.107 \pm 0.051$

Table 3: *Critical exponents for the magnetization β, specific heat capacity α and α', and susceptibility γ and γ'. All data was calculated for $L = 8$ using the Wolff algorithm.*

We did not encounter significant differences when determining the critical exponents using data from the Adaptive Metropolis algorithm. For the same fitting range, we calculated $\beta = 0.380 \pm 0.045$, $\gamma = 1.215 \pm 0.080$ and $\gamma' = 1.142 \pm 0.085$, and $\alpha = -0.119 \pm 0.066$ and $\alpha' = -0.919 \pm 0.088$, which is in agreement with the values in Table 3. In comparison to the study by Holm and Janke,[10] we find that our critical exponents for the order parameter β match well with their prediction ($\beta_{\text{lit}} = 0.362 \pm 0.004$). Similarly, they find $\alpha_{\text{lit}} = -0.112 \pm 0.018$, which matches well with our estimate for α. However, our result for α' in the low-temperature phase does not match the expected value. One reason is the smaller number of data-points for $T < T_C$ compared to $T > T_C$. Holm and Janke further provide a reference value for $\gamma_{\text{lit}} = 1.389 \pm 0.014$. Our estimate of $\gamma = 1.107 \pm 0.051$ is close to the literature, although not in agreement within the errors. This result has the lowest relative error, perhaps underestimating the fitting error. It is important to note that this calculation can be enhanced by performing additional simulations close to T_C, and also by using more Monte Carlo samples for each data points.

[6] This means that we fit $X(T) \sim t^{\pi}$ with $t = (T - T_C)/T_C$.

iv. Comparison of Algorithms

Due to the different convergence behaviours, one might be interested in interchanging the algorithms at different temperatures. In Fig. 12, we show the relative deviation of M and χ when calculated with the three different algorithms (M: Metropolis, Adp. M: Adaptive Metropolis, W: Wolff), normalized by the average value at each temperature. The errors were calculated from the data in Fig. 5 based on an uncertainty propagation of independent variables.

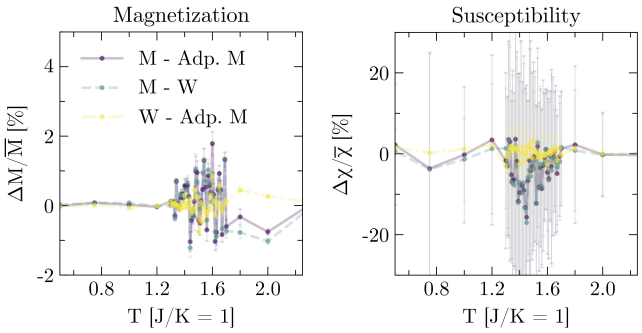

Figure 12: *Deviation of M (left) and χ (right) for the three different algorithms (M: Metropolis, Adp. M: Adaptive Metropolis, W: Wolff), normalized by the average value at each temperature.*

We observe that for M and χ, the deviations are more pronounced close to the critical temperature. This is due to the large fluctuations of M (or equivalently, the diverging susceptibility) close to T_C. Our data shows that the differences between the algorithms are negligible in most cases. This means that the results from different algorithms are consistent with each other. However, one must be careful when using algorithms which show unsatisfactory convergence behaviour at a certain temperature, e.g., the Metropolis algorithm close to T_C in comparison to Wolff algorithm, see also Fig. 2. In Table 4, we present the relative deviations in magnetization between all three algorithms at different temperatures in absolute values (without signs). We see that the calculated values of M do not differ significantly at low and high temperatures. Thus, we can combine the results from different algorithms in those temperature regimes without significantly enhancing the

error. Close to T_C, the use of Metropolis algorithm (M) is discouraged. Instead, we recommend using the Adaptive Metropolis (Adp. M) or Wolff (W) algorithms.

Algorithm:	M - Adp. M	M - W	W - Adp. M
T = 0.01	0.00104 ± 0.00007	0.040051 ± 0.000010	−0.03901 ± 0.00007
T = 0.1	0.0003 ± 0.0007	0.03374 ± 0.00012	0.0334 ± 0.0006
T = 1.0	0.024 ± 0.010	0.078 ± 0.006	0.054 ± 0.008
T = 1.4	0.57 ± 0.19	0.28 ± 0.16	0.30 ± 0.10
T = 1.44	1.04 ± 0.28	1.22 ± 0.27	0.18 ± 0.06
T = 1.5	0.42 ± 0.35	0.33 ± 0.29	0.75 ± 0.20
T = 2.0	0.76 ± 0.09	1.02 ± 0.09	0.27 ± 0.04
T = 10	0.007 ± 0.010	0.055 ± 0.006	0.048 ± 0.009
T = 100	0.0296 ± 0.0011	0.7099 ± 0.0004	0.6803 ± 0.0011

Table 4: *Relative deviations in magnetization* $|\Delta M/\overline{M}|$ *in percent (absolute values) between the different algorithms (M = Metropolis, Adp. M = Adaptive Metropolis, W = Wolff) for different temperatures.*

Another property of interest is the algorithmic performance, which can be determined as the time per algorithmic step. In Fig. 13 (left), we present the time per algorithmic and Monte Carlo step. It is important to note that each step in the Wolff algorithm involves flipping a large cluster of spins, in comparison to single-bond updates for the Metropolis algorithm.

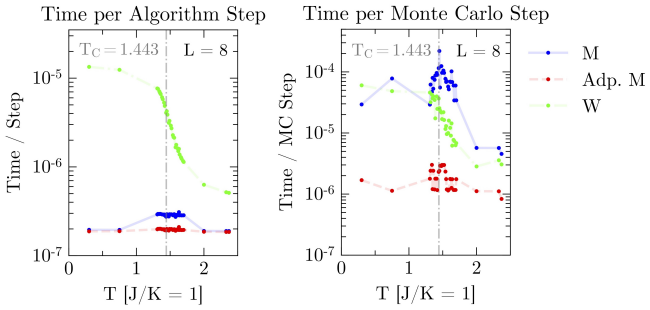

Figure 13: *Comparison of algorithmic performance of the Metropolis (blue), Adaptive Metropolis (red), and Wolff (green) algorithms. Left: Time per algorithmic step. Right: Time per Monte Carlo (MC) step.*

For the Wolff algorithm, can see that the time per step reduces with temperature due to the decreasing cluster size. The Metropolis algorithm becomes slower close to T_C due to a larger number of bond updates. In contrast, the Adaptive Metropolis algorithm shows a constant performance, independent of temperature. This behaviour is expected as the Adaptive Metropolis algorithm leads to a constant acceptance rate, whereas the acceptance rate of the normal Metropolis algorithm only depends on the Boltzmann factor of a certain spin-flip. Similar trends can also be observed in the right plot of Fig. 13, where we present the time per Monte Carlo (MC) step. The data of this figure leads to several interesting conclusions. First, the Metropolis algorithm is always slower than the Adaptive Metropolis algorithm, confirming the assumption that an acceptance rate close to 50 % is desirable for a good algorithmic performance. Secondly, the Wolff algorithm might reduce critical slowing down and thus leads to a smaller number of update steps, but each of those steps takes about one to two orders of magnitude longer, see left plot. Hence, the main benefit of a cluster update procedure might only be beneficious for lattices larger than $L = 8$. Thirdly, as the cluster size shrinks with increasing temperature, the performance differences of all algorithms become negligible for high temperatures $T > T_C$. At least for small lattices, we observe that the Adaptive Metropolis algorithm performs best, independent of temperature. In Fig. 14, we also show that the time per algorithmic step is almost independent of lattice size for the Adaptive Metropolis algorithm. The largest time/step is $1.73 \cdot 10^{-7}$ sec. for L = 2, while the smallest value is $4.37 \cdot 10^{-8}$ sec. for L = 6. While we have no explanation why the performance is roughly one order of magnitude higher for $L = 6$ and $L = 10$, we conjecture that in general, the performance is independent of L. Since the time per algorithmic step is also almost independent of T, see Fig. 13, the previously calculated values universally characterize the algorithmic performance.

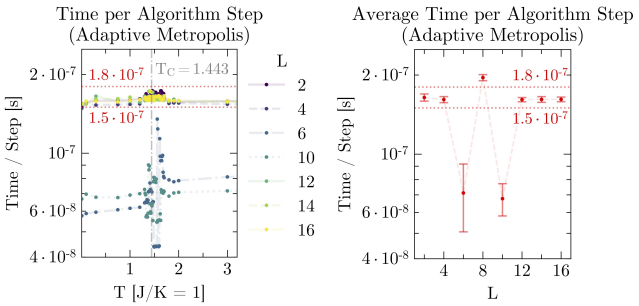

Figure 14: *Time per algorithmic step for the Adaptive Metropolis algorithm and varying lattice lengths* $L = 2, 4, 6, 10, 12, 14, 16$. *Left: Temperature-dependence of time / step. Dotted grey line:* T_C. *Right: Average time / step for all lattice sizes. The errors are standard deviations over the full temperature range, see left plot.*

From the previous data, we can calculate the average time per algorithmic step over all temperatures for all algorithms, see Table 5. Note that those averages are only meaningful for characterizing the performance for critical phenomena, i.e., when applying Monte Carlo sampling around T_C.

This data matches the previous assumption that overall, the Adaptive Metropolis algorithm is best suited for Monte Carlo sampling.

Algorithm:	**Metropolis**	**Adp. Metropolis**	**Wolff**
Time / Step	$(2.74 \pm 0.35) \cdot 10^{-7}$	$(1.96 \pm 0.05) \cdot 10^{-7}$	$(3.8 \pm 3.1) \cdot 10^{-6}$
Time / MC Step	$(7 \pm 4) \cdot 10^{-5}$	$(1.8 \pm 0.7) \cdot 10^{-6}$	$(2.1 \pm 1.5) \cdot 10^{-5}$

Table 5: *Average time per algorithmic step and MC step for all algorithms for $L = 8$ in units of $[s]$. The average value and error were calculated over all different temperatures.*

v. External Field and Magnetic Anisotropy

Our implementation of the external magnetic field h and a magnetic anisotropy k allows us to study symmetry breaking in more detail. In Fig. 15, we show how fast the lattice configurations are de-magnezited, depending on the temperature. Additionally, we have included magnetic anisotropies anti-parrallel to the initial lattice configuration, see right plot.

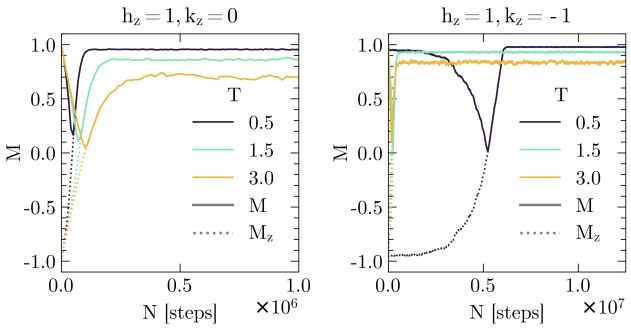

Figure 15: *Implementation of an external magnetic field and a magnetic anisotropy: Magnetization M (solid line) and M_z (dotted line) vs. run-time of Metropolis Algorithm (small step move) for different temperatures $T = 0.5, 1.5, 3.0$ with magnetic field $h_z = 1$. The initial lattice configuration was always chosen to be $M_z = -1$. Left: Up to 10^6 steps without magnetic anisotropy: $k_z = 0$. Right: Up to 10^7 steps with magnetic anisotropy $k_z = -1$, easy axis along $-z$-direction, corresponding to the initial magnetization.*

In the left plot, we see that the de-magnetization is temperature-dependent. For higher temperatures, we need more algorithmic steps to reach thermal equilibrium and align our spin parallel to h after flipping all spins by 180° from their initial value. Moreover, we see that for low temperatures, the magnetization aligns with the external magnetic field ($M \to 1$), as expected, while for higher temperatures, thermal fluctuations reduce the average magnetization, even with an external magnetic field. In the right plot, we can clearly see that a magnetic anisotropy of same magnitude and antiparallel to the external magnetic field leads to a competition between h to align all spins along $h = +z$ and the preferred direction of k along $-z$. Thus, the magnetic anisotropy slows down the de-magnetization (or anti-magnetization, as we see a 180° flip of the magnetization component M_z). Additionally, including a magnetic anisotropy changes the equilibrium value of M as thermal fluctuations are lower compares to $k = 0$.

5. Summary and Outlook

Our implementation of the Heisenberg model in C++ offers a suite of methods to study all physical quantities of interests. We allow to measure and plot E, M, c_V, χ for different temperatures T, side lengths L_x, L_y, L_z and thus dimensions D, interaction strengths J, external magnetic fields $\vec{h}$, and internal magnetic anisotropies $\vec{k}$. Our implementation of the Metropolis algorithm also allows for different trial moves, including Gaussian, small-step and random sampling as well as perfect spin-flips. By measuring physical quantities for different run-times or numbers of algorithmic steps when varying the external parameters, we can also study the convergence behaviours of the (Adaptive) Metropolis and Wolff algorithms. We provide a reference for the number of steps needed to reach thermal equilibrium (non-linear correlation time) and the number of steps for Monte Carlo sampling (linear correlation time). We have showed that the calculation of observables does not depend on the specific algorithm. Hence, we can combine the results of the Metropolis algorithm and Wolff algorithms. Furthermore, we show that the algorithmic time per step is independent of temperature for the Adaptive Metropolis algorithm, while we observe an increase around the critical temperature for the normal Metropolis algorithm, and a decrease with temperature for the Wolff algorithm due to the formation of smaller clusters. We present a conversion factor of Monte Carlo steps to real-life run-time to estimate the total time needed to calculate expectation values. With those tools, we can determine the phase transition, critical temperature, and critical exponents. For those investigations, we simulated lattices between $L = 2$ and $L = 16$ at over 50 temperature points. Our implementation of external fields and internal anisotropies allows us to study the magnetization over time with the Metropolis algorithm, for example as a comparison to experimental data.

Further ideas include the efficient calculation of different types of local magnetic impurities, e.g., by allowing to choose a superposition of local $\vec{k}$, and global anisotropies in the lattice by choosing J to be dependent on the bond orientation. Additionally, including higher dimensions $D \geq 4$ is of interest to use our implementation for the study of higher-dimensional space-time models. One may also be tempted to include other lattices such as triangular, honeycomb, or Kagome lattices. The algorithms can be improved further by implementing parallelization procedures which exist for both the Metropolis [20] and Wolff algorithm. [21] An IPPL implementation would allow to investigate larger lattices on high-performance computing clusters. In order to study external fields at all temperatures efficiently, we are working on an extension of the Wolff algorithm to include both nearest-neighbor interactions as well as interactions with external parameters. Our approach makes use of so-called ghost cells where additional operations of a certain symmetry group on our lattice spins represent magnetic interactions. [22] Further potential for investigation also exists in the study of hybrid Monte Carlo algorithms. For example, it was shown that the addition of Metropolis spin-flips to Wolff cluster algorithms can lead to improved performance. [22,23] The idea to combine the (Adaptive) Metropolis and Wolff algorithms aligns with our results on algorithmic performance, and might outperform our current implementations.

Acknowledgements

We thank K. Sim and A. Adelmann for fruitful discussions. All numerical simulations were performed on the Euler cluster operated by the High Performance Computing group at ETH Zürich.

Code Availability

All code is available on `https://github.com/daniel-schwarzenbach/CSP_Project`.

Contributions

M.C.W. implemented the observables, J.G. implemented the normal and Adaptive Metropolis algorithms, C.T. implemented the Wolff algorithm, D.S. and J.G. implemented the lattice and spin classes. J.G. and C.T. implemented and performed the data calculation. D.S. set up and managed the git-repository. M.C.W., J.G. and C.T. planned the calculations, analysed the data, and created the figures. J.G. and C.T. calculated the linear and non-linear correlation times, determined the critical exponents, and measured the algorithmic performance. D.S. assigned calculations on the EULER cluster. M.C.W. drafted and wrote the report with input from all authors. All authors contributed equally to the theory and final presentations.

July 8, 2024: Note on the second, extended version of this report: M.C.W., J.G. and C.T. determined the non-linear and linear correlation times, measured the algorithmic performance (time per step), compared the algorithms, determined all critical exponents, analyzed finite-size scaling of the susceptibility, and rewrote the report.

Appendix

Non-Linear and Linear Correlation Time

In order to determine the expectation values in eq. 21, we perform M different runs where we use a certain algorithm on the same initial lattice configuration at the same temperature T and measure X every time after a certain amount of steps until we have reached $N_{\max}$ steps. Thus, we get M different arrays of values $[X_i(t_0), ..., X_i(t_{N_{\max}})], i \in \{1, ..., M\}$. We can then easily calculate:

$$\begin{aligned} \langle X(t') \rangle &= \frac{1}{M} \sum_{i=1}^{M} X_i(t') \\ \langle X(t_0) \rangle &= \frac{1}{M} \sum_{i=1}^{M} X_i(t_0) \\ \langle X(t \to \infty) \rangle &= \frac{1}{M} \sum_{i=1}^{M} X_i(t_{N_{\max}}) \,. \end{aligned} \tag{46}$$

Furthermore, we can determine the covariance:

$$\langle X(t_0) X(t') \rangle = \frac{\sum_{i=1}^{M} \big(X_i(t_0) - \langle X(t_0) \rangle \big) \big(X_i(t') - \langle X(t') \rangle \big)}{M}, \tag{47}$$

given the expectation values $\langle X(t_0) \rangle$ and $\langle X(t') \rangle$ at t_0 and at a certain time t', respectively.

Fluctuation-Dissipation Theorem

In general, the **specific heat** is defined as:

$$c_V = \frac{\mathrm{d} \langle E \rangle}{\mathrm{d}T} \tag{48}$$

and the **magnetic susceptibility** is given by:

$$\chi = \frac{\mathrm{d} \langle M \rangle}{\mathrm{d}H}, \tag{49}$$

where H is the external field. For the Ising model, we can easily calculate $\langle E \rangle$ and $\langle V \rangle$:

$$\langle E \rangle = \sum_i p_i E_i = \sum_i \frac{e^{-\beta E_i}}{Z} E_i = \frac{\sum_i E_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}} \tag{50}$$

where

$$\begin{aligned} c_V = \frac{\mathrm{d} \langle E \rangle}{\mathrm{d}T} &= \underbrace{\frac{\mathrm{d}\beta}{\mathrm{d}T}}_{=-1/k_{\mathrm{B}}T^2} \frac{\mathrm{d} \langle E \rangle}{\mathrm{d}\beta} \\ &= -\frac{1}{k_{\mathrm{B}} T^2} \frac{\mathrm{d}}{\mathrm{d}\beta} \left(\frac{\sum_i E_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}} \right) \\ &= -\frac{1}{k_{\mathrm{B}} T^2} \left(\underbrace{\frac{\sum_i E_i^2 e^{-\beta E_i}}{\sum_i e^{-\beta E_i}}}_{=\langle E^2 \rangle} + \underbrace{\left(\frac{\sum_i E_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}} \right)^2}_{=\langle E \rangle^2} \right). \end{aligned} \tag{51}$$

The calculation follows analogously for the magnetic susceptibility χ. For the Heisenberg model, we get the same result, although the derivation is not as clear. If we know how $\langle E \rangle$ depends on T, we have another way to determine c_V. Using eq. 48, we can calculate this derivative, for example by using numerical differentiation.

References

[1] Fogedby, H C.: Solitons and magnons in the classical Heisenberg chain. In: *Journal of Physics A: Mathematical and General* 13 (1980), Nr. 4, 1467. http://dx.doi.org/10.1088/0305-4470/13/4/035. – DOI 10.1088/0305–4470/13/4/035. ISBN 0305–4470

[2] Nutakki, Rajah P. ; Jaubert, Ludovic D. C. ; Pollet, Lode: The classical Heisenberg model on the centred pyrochlore lattice. In: *SciPost Phys.* 15 (2023), 040. http://dx.doi.org/10.21468/SciPostPhys.15.2.040. – DOI 10.21468/SciPostPhys.15.2.040

[3] Chandra, P. ; Coleman, P. ; Larkin, A. I.: Ising transition in frustrated Heisenberg models. In: *Phys. Rev. Lett.* 64 (1990), Jan, 88–91. http://dx.doi.org/10.1103/PhysRevLett.64.88. – DOI 10.1103/PhysRevLett.64.88

[4] Jolicoeur, Th. ; Dagotto, E. ; Gagliano, E. ; Bacci, S.: Ground-state properties of the S=1/2 Heisenberg antiferromagnet on a triangular lattice. In: *Phys. Rev. B* 42 (1990), Sep, 4800–4803. http://dx.doi.org/10.1103/PhysRevB.42.4800. – DOI 10.1103/PhysRevB.42.4800

[5] Pitts, Jackson ; Buessen, Finn L. ; Moessner, Roderich ; Trebst, Simon ; Shtengel, Kirill: Order by disorder in classical kagome antiferromagnets with chiral interactions. In: *Phys. Rev. Res.* 4 (2022), Oct, 043019. http://dx.doi.org/10.1103/PhysRevResearch.4.043019. – DOI 10.1103/PhysRevResearch.4.043019

[6] Joyce, G. S.: Classical Heisenberg Model. In: *Phys. Rev.* 155 (1967), Mar, 478–491. http://dx.doi.org/10.1103/PhysRev.155.478. – DOI 10.1103/PhysRev.155.478

[7] In: Böttcher, Lucas ; Herrmann, Hans J.: *Phase Transitions*. Cambridge University Press, 2021, S. 85â92

[8] Marinkovic, Marina: *Lecture notes on Computational Statistical Physics*. FS 2023

[9] Wolff, Ulli: Critical slowing down. In: *Nuclear Physics B - Proceedings Supplements* 17 (1990), 93-102. http://dx.doi.org/https://doi.org/10.1016/0920-5632(90)90224-I. – DOI https://doi.org/10.1016/0920–5632(90)90224–I. – ISSN 0920–5632

[10] Holm, Christian ; Janke, Wolfhard: Critical exponents of the classical three-dimensional Heisenberg model: A single-cluster Monte Carlo study. In: *Phys. Rev. B* 48 (1993), Jul, 936–950. http://dx.doi.org/10.1103/PhysRevB.48.936. – DOI 10.1103/PhysRevB.48.936

[11] Ruge, C. ; Zhu, P. ; Wagner, F.: Correlation function in Ising models. In: *Physica A: Statistical Mechanics and its Applications* 209 (1994), Nr. 3, 431-443. http://dx.doi.org/https://doi.org/10.1016/0378-4371(94)90195-3. – DOI https://doi.org/10.1016/0378–4371(94)90195–3. – ISSN 0378–4371

[12] Zhou, Zongzheng ; Yang, Ji ; Deng, Youjin ; Ziff, Robert M.: Shortest-path fractal dimension for percolation in two and three dimensions. In: *Phys. Rev. E* 86 (2012), Dec, 061101. http://dx.doi.org/10.1103/PhysRevE.86.061101. – DOI 10.1103/PhysRevE.86.061101

[13] Peczak, P. ; Ferrenberg, Alan M. ; Landau, D. P.: High-accuracy Monte Carlo study of the three-dimensional classical Heisenberg ferromagnet. In: *Phys. Rev. B* 43 (1991), Mar, 6087–6093. http://dx.doi.org/10.1103/PhysRevB.43.6087. – DOI 10.1103/PhysRevB.43.6087

[14] Nguyen, Phong H. ; Boninsegni, Massimo: Superfluid Transition and Specific Heat of the 2D x-y Model: Monte Carlo Simulation. In: *Applied Sciences* 11 (2021), Nr. 11. http://dx.doi.org/10.3390/app11114931. – DOI 10.3390/app11114931. – ISSN 2076–3417

[15] Alzate-Cardona, D; Evans R F L; Restrepo-Parra E. J D; Sabogal-Suárez D. J D; Sabogal-Suárez: Optimal phase space sampling for Monte Carlo simulations of Heisenberg spin systems. In: *J. Phys.: Condens. Matter* 31 (2019), Jan

[16] Metropolis, Nicholas ; Rosenbluth, Arianna W. ; Rosenbluth, Marshall N. ; Teller, Augusta H. ; Teller, Edward: Equation of State Calculations by Fast Computing Machines. In: *The Journal of Chemical Physics* 21 (1953), 06, Nr. 6, 1087-1092. http://dx.doi.org/10.1063/1.1699114. – DOI 10.1063/1.1699114. – ISSN 0021–9606

[17] Haario, Heikki ; Saksman, Eero ; Tamminen, Johanna: An Adaptive Metropolis Algorithm. In: *Bernoulli* 7 (2001), Nr. 2, 223–242. http://www.jstor.org/stable/3318737. – ISSN 13507265

[18] In: Krauth, Werner: *Algorithms and Computations*. Oxford University Press, 2006

[19] In: Barone, Enzo; Organtini Giovanni; Ricci-Tersenghi F. Luciano Maria; Marinari M. Luciano Maria; Marinari: *C-language, Algorithms and Models in Science*. World Scientific, 2013

[20] Calderhead, Ben: A general construction for parallelizing MetropolisâHastings algorithms. In: *Proceedings of the National Academy of Sciences* 111 (2014), Nr. 49, S. 17408–17413. http://dx.doi.org/10.1073/pnas.1408184111. – DOI 10.1073/pnas.1408184111

[21] Kaupužs, Rimšāns J. J. ; Melnik, R. V. N.: Parallelization of the Wolff single-cluster algorithm. In: *Phys. Rev. E* 81 (2010), Feb, S. 026701. http://dx.doi.org/10.1103/PhysRevE.81.026701. – DOI 10.1103/PhysRevE.81.026701

[22] Plascak, J. A. ; Ferrenberg, Alan M. ; Landau, D. P.: Cluster hybrid Monte Carlo simulation algorithms. In: *Phys. Rev. E* 65 (2002), Jun, 066702. http://dx.doi.org/10.1103/PhysRevE.65.066702. – DOI 10.1103/PhysRevE.65.066702

[23] Ostmeyer, Johann ; Berkowitz, Evan ; Luu, Thomas ; Petschlies, Marcus ; Pittler, Ferenc: The Ising model with Hybrid Monte Carlo. In: *Computer Physics Communications* 265 (2021), S. 107978. http://dx.doi.org/https://doi.org/10.1016/j.cpc.2021.107978. – DOI https://doi.org/10.1016/j.cpc.2021.107978. – ISSN 0010–4655

Declaration of Originality

We confirm that we authored the work in question independently and in our own words, i.e. that no one helped us to author it. Suggestions from the supervisor regarding language and content are excepted. We used no generative artificial intelligence technologies.

Mateo Cárdenes Wuttig

Justus Grabowsky

Daniel Schwarzenbach

Constança Tropa

Zürich, July 8th, 2024.