Heisenberg Model in 3D

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

May 7, 2024

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, 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. Moreover, we include a variable interaction strength, an external magnetic field, and local and global magnetic anisotropies. With this, we can study phase transitions and critical phenomena for varying temperatures and cubic lattice sizes up to $L = 32$ for different dimensions $D = 1, 2, 3$ with individual lattice lengths L_i. All of our calculations were performed in parallel on the Euler cluster.

Contents

Lattice models – discrete geometrical structures which originated as a description of atoms in a solid state physics – are one of the most important simplifications in many areas of physics, including condensed matter physics, biophysics, or quantum field theory. 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 such as electrons and holes. Hence, we are interested in determining changes in lattice configurations upon external influences such as temperature or magnetic fields. In this project, we are investigating the **Heisenberg Model**, a lattice model from statistical physics used to model ferromagnetism.

In the first chapter (Background), we provide an introduction into fundamental lattice models such as the Ising Model and Heisenberg Model. Additionally, we explain the phenomenon of phase transitions determined by their order parameter, and introduce critical exponents for physical observables. Moreover, we introduce several quantities 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. In the second chapter on the Implementation, we present the structure of our code and explain how the respective observables are calculated and in which manner we compare the algorithms. Our Results verify the convergence and thermalization of lattices for different temperatures. Then, we show that the expectation values of energy and magnetization do not depend on the respective algorithm we choose. Thus, we can compare the physical quantities regardless of the specific algorithm used. Next, we present the run-times of our benchmark test and compare the algorithmic speeds on different lattices. Our study on phase transitions shows that we can precisely determine the critical temperature and all critical exponents with our implementation. Furthermore, 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 ordering with magnetic fields and internal anisotropies. In the last chapter Summary and Outlook, we resume our findings and provide ideas for future investigations.

I. Background

i. Introduction to Lattice Models

In 1920, Wilhelm Lenz invented the famous **Ising Model** which was solved by his student Ernst Ising in 1925 and nowadays serves as the easiest lattice model in physics. [1] The Hamiltonian H[1] determines the **energy** E of the lattice Λ for a given spin configuration $\boldsymbol{\sigma}$:

$$H(\boldsymbol{\sigma}) = -J \sum_{\langle i,j \rangle} \sigma_i \sigma_j - h \sum_i \sigma_i \tag{1}$$

where the spins on each site can take two opposite values: $\sigma_i = \pm 1 \, \forall i \in \Lambda$. In eq. 1, 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 σ_i with an external magnetic field h. This model can be used to study **phase transitions**. In the Ferromagnetic Ising 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:

$$\boldsymbol{\sigma}_{\text{GS}} = \{\sigma_i = +1 \, \forall i\} \quad \text{or} \quad \boldsymbol{\sigma}_{\text{GS}'} = \{\sigma_i = -1 \, \forall i\} \, . \tag{2}$$

For $h = 0$, there exist two ground states, which corresponds to a $\mathbb{Z}_2$-symmetry. For $h \neq 0$, those two ground states have different energies and the $\mathbb{Z}_2$-symmetry is externally broken. For low temperatures, we expect to find the system in a state with most of the spins parallel to each other, while at high temperature thermal fluctuations dominate which lead to a random orientation of the spins. Thus, we predict two **phases**: a ferromagnetic phase and a paramagnetic phase. To classify those regions, we introduce the **magnetization**:

$$M = \frac{1}{N} \sum_{i=1}^{N} \sigma_i \, . \tag{3}$$

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

M is the average spin orientation over all lattice sites N. In the Ferromagnetic Ising Model, 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 temperature at which the phase transition occurs as T_C.

ii. Heisenberg Model

The natural generalization of the Ising Model is to allow each spin to rotate freely. Instead of $\sigma_i = \pm 1$, we have a unit vector $\vec{\sigma}_i$ at every lattice site i. The Hamiltonian of the so-called **Heisenberg Model** is given by:

$$H_{\text{Heisenberg}} = -J \sum_{\langle i,j \rangle} \vec{\sigma}_i \cdot \vec{\sigma}_j - \vec{h} \cdot \sum_i \vec{\sigma}_i \tag{4}$$

where the scalar product $\vec{\sigma}_i \cdot \vec{\sigma}_j$ between two vectors describes how much those spins overlap. Additionally, we can also provide an external magnetic field as a vector $\vec{h}$. Similar to the Ising Model, we have to sum over all nearest neighbors, i. e. 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. 4 can be computed as following:

$$\sum_i \vec{\sigma}_i = \sum_{i=1}^{L_x} \sum_{j=1}^{L_y} \sum_{k=1}^{L_z} \vec{\sigma}_{i,j,k} \tag{5}$$

$$\sum_{\langle i,j \rangle} \vec{\sigma}_i \cdot \vec{\sigma}_j = \underbrace{\sum_{i=1}^{L_x - 1} \sum_{j=1}^{L_y} \sum_{k=1}^{L_z} \vec{\sigma}_{i,j,k} \cdot \vec{\sigma}_{i+1,j,k}}_{\text{all bonds parallel to x}} + \text{etc.} \tag{6}$$

where (i, j, k) are the Cartesian coordinates of each spin in x-, y-, and z-direction, respectively. In eq. 6, 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. 4 using the summations in eq. 5 and 6, we can efficiently compute the energy of the Heisenberg model defined on a cubic lattice. Similar to the Ising Model, we can choose the magnetization as our order parameter:

$$\vec{M} = \frac{1}{N} \sum_i \vec{\sigma}_i \tag{7}$$

where $N = L_x \cdot L_y \cdot L_z$ is the total number of spins. Again, we will also expect a phase transition at a certain critical temperature T_C below which all spins are preferably aligned parallel to each other. One key difference to the Ising Model is the absence of the discrete $\mathbb{Z}_2$-symmetry. Instead, we observe that the total energy does not change if we rotate all spins by the same rotation matrix. This naturally leads to a continuous $O(3)$-symmetry for the 3d Heisenberg Model without external field $\vec{h} = 0$. If $\vec{h} \neq 0$, this symmetry is externally broken. Again, we expect $M \to 1$ for $T \to 0$ and $M \to 0$ for $T \to \infty$ where $M = \|\vec{M}\|$. 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 quasi-particles. 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. [2] As such, classical models are used to simulate features such as frustration [3,4] or perform calculations on interesting systems such as different lattices. [5,6]

iii. Partition Sum and Expectation Values

In order to perform calculations on the Heisenberg Model, one needs to compute expectation values of quantities such as the energy or magnetization. In general, we define the **expectation value** of an observable X as follows:

$$\langle X \rangle = \sum_i p_i X_i \,. \tag{8}$$

Here, p_i is the probability that the observable 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:

$$Z = \sum_i e^{-\beta E_i} \tag{9}$$

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

$$P_i = \frac{1}{Z} e^{-\beta E_i} \,. \tag{10}$$

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

$$\langle E \rangle = -\frac{\partial \log Z}{\partial \beta} \quad \text{with} \quad Z = Z(\beta) \,. \tag{11}$$

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. For an infinite number of microstates, we have:

$$Z = \frac{1}{h^3} \int \mathrm{d}^3 q \, \mathrm{d}^3 p \, e^{-\beta \mathcal{H}(q,p)} \tag{12}$$

where the energy is given by the Hamiltonian of a certain phase space point (q, p). For the Heisenberg Model, we have:

$$Z_{\text{Heisenberg}} = \int_{\Omega_1} \cdots \int_{\Omega_N} \prod_{i=1}^{N} \frac{\mathrm{d}\Omega_i}{4\pi} e^{-\beta \mathcal{H}} \tag{13}$$

where we have a product over the solid angles of each spin. [7] Again, we can only compute $Z_{\text{Heisenberg}}$ exactly for small systems. This explains why we need numerical algorithms to simulate this lattice.

iv. Monte Carlo Phase Space Sampling

In the previous section, we explain why the use partition sums Z which cover the whole phase space is generally not suitable to calculate expectation values of larger systems. Instead, 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. For sufficiently large N, we can estimate the expectation values of observables X such as energy E and magnetization M[3] as follows:

$$\langle X \rangle = \frac{1}{N} \sum_{i=1}^{N} X_i \tag{14}$$

and

$$\langle X^2 \rangle = \frac{1}{N} \sum_{i=1}^{N} X_i^2 \tag{15}$$

under the assumptions that:

(a) $p_i = \frac{1}{N}$, i. e. the probability of 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.

[2] If we consider the Ising Model with two possible states at every site, 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.

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

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 steads state value $X(t=\infty)$ [8]

$$\phi_{X,\mathrm{nl}}(t) = \frac{\langle X(t)\rangle - \langle X(t\to\infty)\rangle}{\langle X(t_0)\rangle - \langle X(t\to\infty)\rangle}, \tag{16}$$

where t_0 is the time at which we have reached thermal equilibrium. We see that

$$\phi_{X,\mathrm{nl}}(t_0) = 1 \quad \text{and} \quad \phi_{X,\mathrm{nl}}(t\to\infty)\to 0\,. \tag{17}$$

One idea to determine thermal equilibrium is to determine c_V, 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 measure the non-linear correlation directly. We expect that:

$$\phi_{X,\mathrm{nl}}(t) \sim e^{-t/t_0^{\mathrm{nl}}} \tag{18}$$

where t_0^{nl} is the **non-linear correlation time**, which is also defined as: [8]

$$t_0^{\mathrm{nl}} = \int_0^\infty \mathrm{d}t'\,\phi_{X,\mathrm{nl}}(t') \tag{19}$$

as we know that:

$$\int_0^\infty \mathrm{d}t\, e^{-t/\tau} = \tau\,. \tag{20}$$

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

$$\tau \approx \frac{1}{2} + \sum_{i=1}^{N_{\mathrm{cutoff}}} \phi_i \tag{21}$$

where $\phi_0 = 1$. [9] As the correlations increase close to the critical temperature, we can expect t_0^{nl} to increase when approaching T_C. This is know as the **critical slowing down** of the dynamics: [10]

$$t_0^{\mathrm{nl}} \sim \lim_{T\to T_C} |T-T_C|^{-z} \tag{22}$$

where z is the non-linear dynamical critical exponent. [8] Thus, is becomes increasingly hard to calculate expectation values upon approaching T_C. This power law behavior is due to a critical behaviour general to the Heisenberg Model, see also the section on phase transitions and critical exponents. Moreover, we expect the correlation times to depend on the lattice dimensions, since a larger lattice leads to more steps needed to reach thermal equilibrium. Thus, we have to define t_V for a given volume $V=L^3$ and extrapolate it for a different volume $V' = L_x\cdot L_y\cdot L_z$:

$$t_{V'} = \frac{V'}{V} t_V\,. \tag{23}$$

Similar to the previous discussion, we 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:

$$\phi_{X,\mathrm{l}}(t) = \frac{\langle X(t_0)\cdot X(t)\rangle - \langle X(t_0)\rangle^2}{\langle X(t_0)^2\rangle - \langle X(t_0)\rangle^2}, \tag{24}$$

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

$$\phi_{X,\mathrm{l}}(t) \sim e^{-t/t_0^{\mathrm{l}}}\,. \tag{25}$$

Thus, we can determine t_0^{l} with the same approximation, see eq. 21. In order to determine the exception values in eq. 24, we perform M different runs where we use a certain algorithm of 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}. Thus, we get M different arrays of values $[X_i(t_0),\dots,X_i(t_{N_{\mathrm{max}}})], i\in\{1,\dots,M\}$. We can easily calculate:

$$\begin{aligned}\langle X(t')\rangle &= \sum_{i=1}^{M} X_i(t')\\ \langle X(t_0)\rangle &= \sum_{i=1}^{M} X_i(t_0)\\ \langle X(t\to\infty)\rangle &= \sum_{i=1}^{M} X_i(t_{N_{\mathrm{max}}})\end{aligned} \tag{26}$$

Furthermore, we can determine the covariance:

$$\langle X(t_0)X(t')\rangle = \frac{\sum_{i=1}^{M}\big(X_i(t_0)-\langle X(t_0)\rangle\big)\big(X_i(t')-\langle X(t')\rangle\big)}{M} \tag{27}$$

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

v. Fluctuation-Dissipation Theorem

Given a series of measurements $X_1,\dots,X_N$ which can be used to compute expectation values of the respective observables, see eq. 14 and 15, we are also interested in the variance:

$$\sigma_X^2 = \langle X^2\rangle - \langle X\rangle^2\,. \tag{28}$$

The **fluctuation-dissipation theorem** provides a method to use such fluctuations in energy or magnetization and associate them with other quantities such as the specific heat or susceptibility. In particular, the **specific heat** is defined as:

$$c_V = \frac{\mathrm{d}\langle E\rangle}{\mathrm{d}T} \tag{29}$$

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

$$\chi = \frac{\mathrm{d}\langle M\rangle}{\mathrm{d}H} \tag{30}$$

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

$$\langle E\rangle = \sum_i p_i E_i = \sum_i \frac{e^{-\beta E_i}}{Z} E_i = \frac{\sum_i E_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}} \tag{31}$$

where

$$\begin{aligned} c_V = \frac{\mathrm{d}\langle E\rangle}{\mathrm{d}T} &= \underbrace{\frac{\mathrm{d}\beta}{\mathrm{d}T}}_{=-1/k_\mathrm{B}T^2} \frac{\mathrm{d}\langle E\rangle}{\mathrm{d}\beta} \\ &= -\frac{1}{k_\mathrm{B}T^2}\frac{\mathrm{d}}{\mathrm{d}\beta}\left(\frac{\sum_i E_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}}\right) \\ &= -\frac{1}{k_\mathrm{B}T^2}\left(\underbrace{\frac{\sum_i E_i^2 e^{-\beta E_i}}{\sum_i e^{-\beta E_i}}}_{=\langle E^2\rangle} + \underbrace{\left(\frac{\sum_i E_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}}\right)^2}_{=\langle E\rangle^2}\right). \end{aligned} \tag{32}$$

Thus, we can relate variances in energy E to the specific heat c_V if we know the temperature T:

$$c_V = -\frac{1}{k_\mathrm{B}T^2}\left(\langle E^2\rangle - \langle E\rangle^2\right). \tag{33}$$

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

$$\chi = \frac{1}{k_\mathrm{B}T}\left(\langle M^2\rangle - \langle M\rangle^2\right). \tag{34}$$

Note that the magnetic susceptibility is defined even for $H = 0$ (no external field). The calculations work analogously for the Heisenberg Model.

vi. Phase Transitions and Critical Exponents

The phase transition from a ferromagnetic to a paramagnetic phase can be measured by using 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 (i. e. 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 diverges (or becomes maximal with finite size), but there is no jump, in contrast to first order transitions. This criterion (i. e. determining 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**. [11] For example, we are interested in the critical exponent of the order parameter $\beta \in \mathbb{R}$:

$$M(t) \sim \lim_{t\to 0}|t|^{-\beta} \quad \text{with} \quad t := \frac{T - T_C}{T_C} \tag{35}$$

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

$$c_V(t) \sim \lim_{t\to 0}|t|^{-\alpha} \tag{36}$$

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

$$\chi(t) \sim \lim_{t\to 0}|t|^{-\gamma} \tag{37}$$

are defined analogously. [12] Another useful quantity is the **Binder cumulant**:

$$U_L = 1 - \frac{\langle M^4\rangle}{3\langle M^2\rangle^2} \tag{38}$$

which is defined by using $\langle M^2\rangle$ and $\langle M^4\rangle$ to determine T_C precisely since U_L is independent of L at T_C. [8] Hence, we can plot $U_L(T)$ for different system lenghts L (or volumes) against the temperature and determine the critical temperature T_C as the point where the different $U_L(T)$ curves overlap. [8] Even with periodic boundary conditions, we expect the critical temperature T_C to depend on the dimensionality and system size. For example, Holm and Janke report $T_C \approx 1.440$ for a lattice size L of order $\mathcal{O}(10)$, [11] which agrees with the result of Peczak et al. of $T_C \approx 1.4432$. [13] In general, we expect the critical temperature to larger for smaller lattice sizes. [11,14]

vii. Extensions of the Heisenberg Model

Naturally, one may try to extend the (classical) Heisenberg Model to describe lattice systems with additional constraints. 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:

$$H_{\text{Heisenberg}} = -J\sum_{\langle i,j\rangle}\vec{\sigma}_i\cdot\vec{\sigma}_j - \vec{h}\cdot\sum_i\vec{\sigma}_i - \sum_i(\vec{k}\cdot\vec{\sigma}_i)^2\,. \quad (39)$$

Again, we can evaluate eq. 39 efficiently by performing an additional summation over all lattice sites $\sum_i$, analogous to eq. 4. To include **local magnetic impurities**, we can make $\vec{k}$ position dependent:

$$\vec{k}\to\vec{k}_{i,j,k} = \begin{cases}\vec{k}, & \text{if } (i,j,k)\in\Lambda_{\text{impurity}}\\ \vec{0}, & \text{otherwise}\,.\end{cases} \quad (40)$$

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

viii. Algorithms

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

viii.1. Metropolis Algorithm

The **Metropolis Algorithm** was first published in 1953 to simulate equations of state.[16] In the following pseudocode on Algorithm 1, we describe the Metropolis algorithm for a given number of maximal steps $N_{\max}$. 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 $p = e^{-\beta\Delta E}$. However, this leads to critical slowing down near phase transitions as the system takes increasingly longer to reach thermal equilibrium. In order to achieve satisfactory converge behaviour, Haario et al. proposed an **Adaptive Metropolis Algorithm**[17] which is based on

Algorithm 1 Metropolis Algorithm

```
Require: Initial lattice configuration
  N = 1
  while $N < N_{\max}$ do
      Choose random spin at site $i$
      Compute $\Delta E$ 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
```

the Random Walk Metropolis algorithm,[16] see algorithm 1. Here, the *trial move* can be chosen to be a **Gaussian move** instead of a spin flip. This means that the new spin direction $\vec{\sigma}'$ after a flip is given by:

$$\vec{\sigma}' = \frac{\vec{\sigma}+\alpha\vec{\Gamma}}{|\vec{\sigma}+\alpha\vec{\Gamma}|} \quad (41)$$

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:

$$\alpha' = \alpha\cdot f\,,\quad f = \frac{1}{2\cdot(1-R)} \quad (42)$$

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:

$$R_{n\geq 2} = \frac{\sum_{i=1}^{n-1}R_i}{n-1} \quad \text{with} \quad R_i = \begin{cases}1, \text{if accepted}\\ 0, \text{if declined}\end{cases} \quad (43)$$

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. 43. 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 the temperature and anisotropy of the system, leading to lower correlation times and a faster convergence toward 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.41), respectively. Obviously, a perfect 180° spin flip cannot lead to convergence independent of the initial lattice configuration.

viii.2. Wolff Algorithm

The main disadvantage of the Metropolis algorithm is the critical slowing down near a phase transition. The **Wolff Algorithm** [10] updates clusters of sites instead of single spins to overcome this challenge. Hence, long autocorrelation time can be avoided as the converge only shows a weak dependence on the lattice size. However, the use of the Wolff algorithm is discouraged far away from critical points or for small systems where autocorrelation times are negligible for the convergence behaviour. [8] In the following pseudocode on Algorithm 2, we explain the Wolff algorithm implemented by means of a stack. In every 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.

Algorithm 2 Wolff Algorithm

```
Require: Initial lattice configuration
  N = 1
  Stack = {}
  while $N < N_{\max}$ do
      Choose random spin $\sigma_r$ and random lattice site $i$
      Add $i$ to the stack
      Flip $\sigma_i$ w.r.t. the hyperplane orthogonal to $r$:
        $\sigma_i \rightarrow \sigma_i - 2(\sigma_i \cdot r)r$
      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
```

2. Implementation

i. Structure of the Code

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

- $\mathtt{L_x}, \mathtt{L_y}, \mathtt{L_z}$: 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 choose 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. Additionally, we offer several methods, including:

- $\mathtt{L_x}$`()`,$\mathtt{L_y}$`()`,$\mathtt{L_z}$`()`: lattice dimensions,
- `memory_size()`: amount of memory for this lattice,
- `get_total_size()`: total no. of lattice sites.

The following **algorithms** functions:

- `void Metropolis(&lattice, T, time, steps, J, h)`
- `void Metropolis_advanced(&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` = $n_{\mathtt{steps}}$, 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 will lead to different changes in the lattice during the same amount of time. All algorithms also need the interaction strength J, the current magnetic field vector (`h`, with three different components $\mathtt{h_x}$, $\mathtt{h_y}$, $\mathtt{h_z}$), and the anisotropy vector (`k`, again with three components $\mathtt{k_x}$, $\mathtt{k_y}$, $\mathtt{k_z}$) to calculate the energy cost of each spin flip. When calling the respective function, the algorithm changes part of the spin configuration of the lattice through the reference `&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.

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 perform, 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 $N_{\text{max}} = 30$, 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.

For example, for `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:

$$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i\,. \tag{44}$$

Additionally, we have to include weights to correct for the fact that the measurement after 20 steps already includes the observable value at 10 steps. Thus, we calculate:

$$\overline{x}^{\text{w}} = \frac{1}{n \cdot N_{\text{sum}}}\sum_{i=1}^{n} i \cdot N_s \cdot x_i \tag{45}$$

where the sum of all weights is given by

$$N_{\text{sum}} = \sum_{i=1}^{\text{no\_of\_steps}} \texttt{array\_of\_steps}[i] \tag{46}$$

and $i \cdot N_s$ is the total number of steps that have been performed for the i-th step. 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` = $[\sigma_{M,\texttt{steps=10}}, \sigma_{M,\texttt{steps=20}}, \sigma_{M,\texttt{steps=30}}]$. The standard deviation of an array of data points $x_i \in [x_1, ..., x_n]$ with mean value $\overline{x}$ are calculated for the n-th sample as following:

$$\sigma_{X,n} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2}\,. \tag{47}$$

Obviously, $\sigma_{X,1} = 0$. 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 when we calculate the thermalization. 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 $[N_s, 2 \cdot N_s, ..., N_{\text{max}}]$ as follows:

$$\sigma^{\text{w}}_{X,n} = \sqrt{\frac{1}{n \cdot N_{\text{sum}}}\sum_{i=1}^{n} i \cdot N_s \cdot (x_i - \overline{x}^{\text{w}})^2}\,. \tag{48}$$

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 determine the thermalization by checking 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. We can calculate the number of steps precisely by using the non-linear correlation time, see eq. 16. Obviously, we have to check that the mean value $\overline{X}$ converges to the anticipated equilibrium value X_{eq}. Moreover, it is important to ensure that the variance for the i-th data point $\sigma_{i,\text{X}}$ of all values up to a certain average value $\overline{X}_i$ generally

[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.

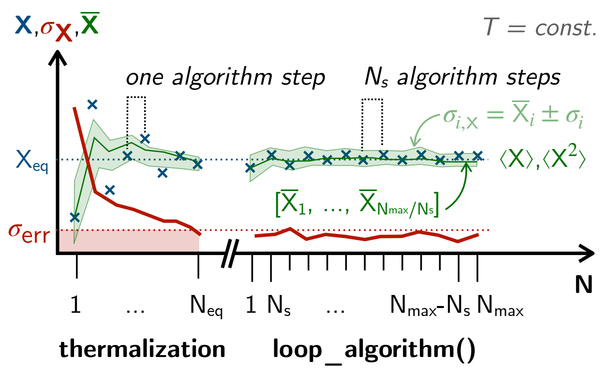

Figure 1: *Measuring the observables: First, we wait for the thermalization to set in after a certain no. of steps N_{eq}. This is checked by plotting a quantity $X = c_V, \chi$ which converges upon reaching thermal equilibrium. Then, the variance of X goes to zero. After reaching thermal equilibrium, we measure the respective observable $X = M, E$ (blue cross) every time after running the algorithm for N_s steps and repeat this until we have performed N_{max} steps. The resulting array of values $[\overline{X}_1, \ldots, \overline{X}_{N_{max}/N_s}]$ is used to determine the expectation values $\langle X \rangle$ (green line) and $\langle X^2 \rangle$ as well as the error (light-green area) up to the i-th step. After N_{max} steps, we can determine the order parameter M, the specific heat c_V, and the magnetic susceptibility χ.*

scales as $\sigma_N \sim \frac{1}{\sqrt{N}}$. With this procedure, we determine a fixed number of steps N_{eq} after which the algorithm has reached thermal equilibrium. We expect N_{eq} to be different for the Metropolis and Wolff algorithms, respectively. Unlike for the Metropolis algorithm, each step in the Wolff algorithm generally consists of a different amount of operations. Hence, the number of steps N of each algorithm is preferable as a condition for thermalization. However, we can repeat the procedure analogously for t instead of N. After choosing a certain time step size δt, we can determine the time t_{eq} after which the respective algorithm has thermalized. The thermalization condition has to be checked 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_{eq}, we will provide 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 measuring expectation values in the thermalized system (after N_{eq} steps) is equivalent to the way how we determine the thermalization in the first place, apart from the specific quantity that we determine.[5] We start by determining the number of steps N_{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, i. e. t_{s}, then we have to ensure that:

$$t_{\text{s}} > \tau_0 \qquad (49)$$

where $\tau_0 = \tau_0^\alpha(T)$ is the **autocorrelation time**, which will also depend on temperature, the lattice volume, and the respective algorithm. We expect $N_{\text{eq}}^\alpha(T)$ to become maximal upon approaching T_C. This phenomenon is known as critical slowing down.[8] After that, we will measure $X = M, E$ every N_S steps until we have performed $N_{\max}$ steps.[6] In $[X_1, \ldots, X_{\texttt{N}_{\texttt{max}}/\texttt{N}_{\texttt{S}}}]$, we have saved `round(`$\texttt{N}_{\texttt{max}}/\texttt{N}_{\texttt{S}}$`)` values of the observable X for every different external condition. The average value of this array is $\langle X \rangle$, and the variance equals σ_X, i. e. the specific heat 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.

iv. Benchmarking

In order to compare the different algorithms and their implementation, we define the following **benchmark test**:

During this test, our lattice first reaches thermal equilibrium after N_{eq} steps. Then, we measure the observables and their variances every N_S steps until $N_{\max}$ is reached. This procedure is repeated $M_{\max}$ times. We can run this benchmark this for different L and T to compare the respective algorithms before and after parallelization to determine the possible speed-up.

[5] $c_V \sim \sigma_E$ for the thermalization and E for `loop_algorithm()`.

[6] When using t_{S} instead of N_{S}, we have to choose a certain time step interval δt and a maximum amount of time $t_{\texttt{max}}$.

Algorithm 3 Benchmark Test

```
Require: Lattice parameters: $L$ and $T$, algorithm
Require: Test parameters: $M_{\max}$, $N_{\text{eq}}$, $N_{\max}$, $N_S$
  $M = 1$
  while $M < M_{\max}$ do
    Thermalize with $N_{\text{eq}}$
    $N = 1$
    while $N < N_{\max}$ do
      Calculate $\langle M \rangle$, $\langle M^2 \rangle$, $\langle E \rangle$, $\langle E^2 \rangle$
      $N += N_S$
    end while
    $M += 1$
  end while
```

3. Results

In the following section, we will always use units of $J = 1$ and $k_B = 1$, thus leading to $\beta = \frac{1}{T}$. 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 used 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 (i. e. variance of the energy) for different algorithms depending on the number of steps.

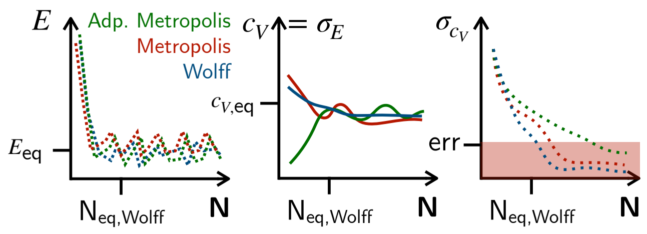

Figure 2: *Measuring the convergence of c_v to determine thermal equilibrium. All data is calculated for $L = x$ and $T = y$.*

For a given lattice size L and temperature T, we determine the energy E after each step. The energy variance is the specific heat c_V which converges to its equilibrium value after a sufficient amount of steps. We plot the variance of c_V and determine an error (`err`) at which we consider the system to be thermalized after a certain amount of steps N_{eq}. We set the error to x. This leads to $N_S^{\text{Metropolis}} = 130$, $N_S^{\text{Adp. Metropolis}} = 110$, and $N_S^{\text{Wolff}} = 10$ for $T = 1$. In Fig. 3, we plot N_{eq} for different temperatures, leading to a different number of steps needed to reach thermal equilibrium. The dotted lines indicate our linear ramps. All of those calculations are performed for $L = 8$.

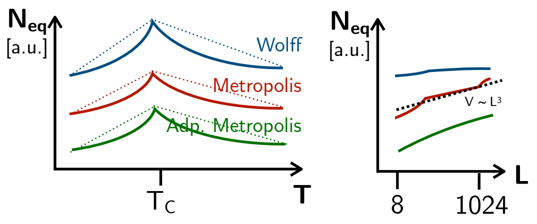

Figure 3: *Left: Number of steps N_{eq} to reach thermal equilibrium for different temperatures, depending on the algorithms. Dottes lines: linear ramp. Right: Dependence of N_{eq} on the system size for different algorithms. Dotted lines: Linear increase with system volume.*

In the left figure, we see the phenomenon of critical slowing down, leading to a higher amount of steps needed to reach thermal equilibrium when using the Metropolis algorithm (up to $\mathcal{O}(10^3)$). The adaptive Metropolis algorithms lead to a slight reduction of steps needed. In contrast, the number of steps only increases by $\mathcal{O}(10^2)$ when using the Wolff algorithm. In the right plot, we show that the amount of steps N_{eq} is almost independent of the lattice size L for the Wolff algorithm, while it shows a linear dependence on the system's volume $V = L^3$ for the Metropolis algorithms, see dotted line.

Next, we have to determine the autocorrelation time where we have used a similar procedure. In Table 1, we present the autocorrelation times (given as number of steps N_S) for different algorithms and temperatures. Here, we used temporal correlations and determined the time $t_{\text{autocorrelation}}$ after which two lattice configurations are uncorrelated by demanting that $\phi_{X,\text{non-linear}}(t_{\text{autocorrelation}}) = 0.01$, see eq. 16.

Algorithm:	**Metropolis**	**Adp. Metropolis**	**Wolff**
$T = 0.1$	0.1	0.1	
$T = 1$			
$T = T_C$			
$T = 100$			
$T = 1000$			

Table 1: *Autocorrelation time for different temperatures and algorithms.*

In Fig. 4, we show that the calculates observables do not differ significantly depending on the algorithm used. We used $N = 1000$ samples for each expectation value for $T \in [0.1, 100]$ with steps of $\Delta T = 0.1$ for a cubic lattice of size $L = 32$. The maximal deviation between the algorithms are also listed in Table 2.

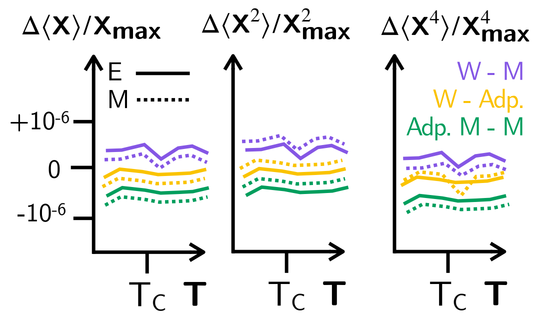

Figure 4: *Differences between the different algorithms (M = Metropolis, Adp. M = Adaptive Metropolis, W = Wolff) in calculating the expectation values $\langle X \rangle$, $\langle X^2 \rangle$ and $\langle X^4 \rangle$ for E (solid lines) and M (dotted lines), normalized by the maximal value of each respective expectation value.*

Algorithm:	**W - M**	**W - Adp. M**	**M - Adp. M**
$\langle M \rangle$	$3.5 \cdot 10^{-6}$	$3.5 \cdot 10^{-6}$	$3.5 \cdot 10^{-6}$
$\langle M^2 \rangle$			
$\langle M^4 \rangle$			
$\langle E \rangle$			
$\langle E^2 \rangle$			
$\langle E^4 \rangle$			

Table 2: *Maximal deviations between the different algorithms (M = Metropolis, Adp. M = Adaptive Metropolis, W = Wolff) for $\langle X \rangle$, $\langle X^2 \rangle$ and $\langle X^4 \rangle$ where $X = M, E$.*

We see that all deviations are order $\mathcal{O}(10^{-4})$ or lower between the algorithms. The highest deviations occur when calculating $\langle M^4 \rangle$, which is expected as M^4 has the highest variance. Thus, we can expect that c_V, χ, T_C and ξ will not depend on the specific algorithm used. Moreover, we can arbitrarily interchange results obtained from different algorithms when calculating the observables.

ii. Benchmarking Space & Time Complexity

Our efforts to parallelize the algorithms are mainly aimed towards:

(a) simulations of larger lattices for $L > 1024$, which is the largest lattice one can save in the usual computer RAM,

(b) precise calculations of observables, for which many similar but uncorrelated simulations are used, e. g. for different temperatures.

We have defined a realistic benchmark test to compare the thermalization and calculation of observables, see Algorithm 3. In Table 3 and **??**, we show the benchmark time for single-core and multi-core runs, respectively. In Table **??**, we also present the speed-up factor in comparison to the single-core run. All calculations were performed under the same conditions on the EULER cluster.

OPTIONAL

single-core	**Metropolis**	**adp. Metropolis**	**Wolff**
L = 8	10.3 s	8.3 s	4.3 s
L = 16	10.2 s		

Table 3: *Benchmark run-time for the single-core implementation at $T = 0.5$*

Our efforts to parallelize the algorithm lead to a speed-up factor of up to x. ETC:

In order to compare the number of steps each algorithm takes, we offer a conversion from N_S to t. For the Metropolis algorithm, the computational time needed for each step is independent of temperature, while for the Wolff algorithm, we notice a small dependence. In Fig. 5, we show that the time per algorithmic step is almost independent of temperature and lattice size for the Metropolis algorithm, with deviations of less than $x\%$. For the Wolff algorithm, we see that the time each step takes depends on the cluster size, thus leading to a relative speed-up for T close to T_C vs. T far away from T_C. Of course, this is compensated by the higher autocorrelation time close to T_C.

We conclude that the conversion of steps N to time t depends on the algorithm. For the Metropolis algorithm, we have:

$$1\,\text{Step} = 0.01\,\text{s} \tag{50}$$

which is independent of temperature. For the Wolff algorithm, we can approximate the conversion factor depend-

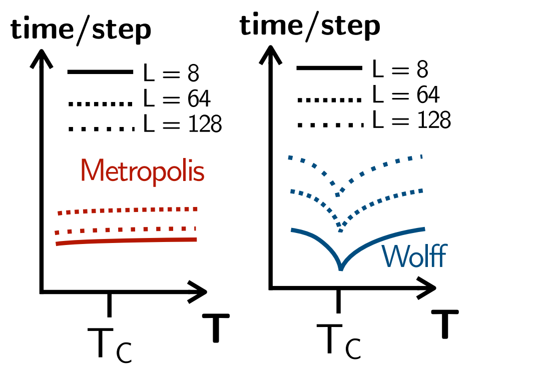

Figure 5: *Average time per algorithmic step for different lattice sizes L, depending on the temperature. Left: Metropolis Algorithm. Right: Wolff Algorithm.*

ing on T as follows:

$$1\,\text{Step} = 0.01 \cdot \frac{|T - T_C|}{31.3}\,\text{s}\,. \tag{51}$$

Convert algorithm steps (temperature-independent for Metropolis, temperature-dependent for Wolff) in time, i. e. determine how long each Metropolis step takes on the EULER cluster and determine how long each Wolff step takes for a given temperature, both given a random lattice each time, and average this over 1000 samples.

iii. Phase Transition and Critical Exponents

One of the most interesting features of the Heisenberg Model is its phase transition from a ferromagnetic low-T phase to a paramagnetic high-T phase. In Fig. 6, we plot the order parameter M and the specific heat capacity c_V for different temperatures. We can clearly recognize a phase transition at $T_C \approx 1.3$

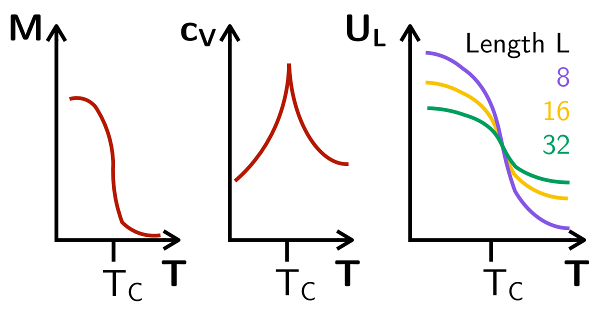

Figure 6: *The magnetization (a) and specific heat capacity (b) for different temperatures indicate a phase transition at T_C. In (c), we plot the binder cumulant for different lattice sizes to determine the critical temperature T_C.*

The critical temperature can be determined precicely by means of the Binder cumulant, see right figure. By calculating U_L as a function of temperature for different lattice sizes L, we can determine the intersection at $T = 1.32 \pm 0.21$. This fits well into the literature values of...

Literature comparison. HOW GOOD IS CRITICAL TEMPERATURE

Next, we are interested in determining the critical exponents. Since we are applying least square fits close to T_C, we will only use the Wolff algorithm to avoid critical slowing down. In Fig. 7, we plot the specific heat c_V, magnetic susceptibility χ, correlation length ξ and magnetization M against temperature for different cubic lattice sizes $L = 4, 8, 16, 32, 64, 128, 256, 512$. We have calculates the respective observables for $T \in [0.1, 10]$ with temperature steps of $\Delta T = 0.01$.

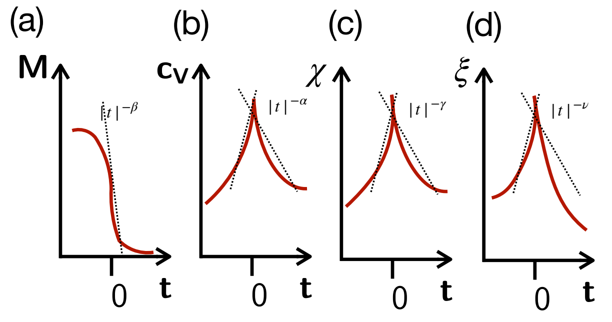

FIGURE 7: *Critical exponents for the magnetization (a), specific heat capacity (b), magnetic susceptibility (c), and correlation length (d).*

In the following TABLE 4, we list the respective critical exponents x, x' with their error $\pm\Delta x$ based on the fitting procedure.

Algorithm:	c_V	M	ξ	χ
$\lim_{t\to 0^+}$	$\alpha = 3.2 \pm 0.1$			
$\lim_{t\to 0^-}$	$\alpha = 3.1 \pm 0.1$			

TABLE 4: *Critical exponents for the specific heat capacity c_V, ...*

COMPARISON WITH LITERATURE

ONLY FOR METROPOLIS Our implementation of the external magnetic field h and a magnetic anisotropy k allows us to study symmetry breaking in more detail. In FIG. 8, we show how fast the lattice configurations are demagnezited, depending on the algorithm. Additionally, we have included magnetic anisotropies anti-parrallel to the initial lattice configuration.

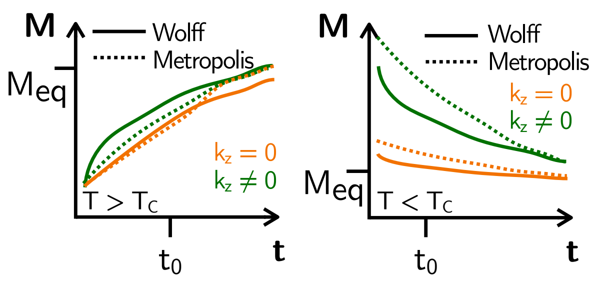

FIGURE 8: *Magnetization vs. run time of Metropolis (solid line) and Wolff (dotted line) algorithms, respectively. Left: High-temperature phase $T > T_C$ where the initial lattice has random orientation. Right: Low-temperature phase $T < T_C$ where the initial lattice is oriented in $+z$-direction. The magnetic anisotropy $|\vec{k}| = 1$ is along $-z$-direction.*

4. SUMMARY AND OUTLOOK

Our implementation of the Heisenberg Model in C++ offers a suite of methods to study all physical quantities of interests such as the correlation length ξ, the critical temperature T_C, and all critical exponents $(\alpha, \alpha', \beta, \gamma, \gamma', \nu, \nu', \eta)$. Moreover, we allow to measure and plot $E, \vec{M}, c_V, \chi, \xi$ 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 $\vec{k}$. For this, we needed to extend the Wolff algorithm to include both nearest-neighbor interactions as well as interactions with external parameters. Our implementation of the Metropolis algorithm allows for different trial moves, including Gaussian and x. By measuring such quantities for different run times or numbers of algorithmic steps when varying the external parameters, we can also study the convergence behaviours of the Metropolis and Wolff algorithms, respectively. We have achieved to parallelized the algorithms and the functions to efficiently investigate lattice systems for different external parameters. This allows us to study lattice sizes up to $L = x$ with a speed-up of x compared to y.

Further ideas for future implementations include the efficient calculation of different types of local magnetic impurities, i. e. 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, respectively. The algorithms can be improved further by xxx.

ACKNOWLEDGEMENTS

We thank K. Sim and A. Adelmann for fruitful discussions. The numerical simulations for the largest lattices 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 algorithm, C.T. implemented the Wolff algorithm, D.S. and J.G. implemented the lattice and spin classes. J.G. and C.T. implemented 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. All authors contributed equally to the theory and final presentations.

References

[1] Ising, Ernst: Beitrag zur Theorie des Ferromagnetismus. In: *Zeitschrift für Physik* 31 (1925), Nr. 1, 253–258. http://dx.doi.org/10.1007/BF02980577. – DOI 10.1007/BF02980577. ISBN 0044–3328

[2] 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

[3] 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

[4] 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

[5] 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

[6] 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

[7] 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

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

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

[10] 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

[11] 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

[12] 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

[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

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, May 7th, 2024.