physics760 - Computational Physics: The Kosterlitz-Thouless phase transition and the XY model

Cedric Hübbel
3306616

Kevin Helders
2993317
(Dated: March 2, 2026)

ABSTRACT HERE

Usage: Secondary publications and information retrieval purposes.

Structure: You may use the `description` environment to structure your abstract; use the optional argument of the `\item` command to give the category of each item.

I. INTRODUCTION

The topic of this report is the Kosterlitz-Thouless phase transition and the XY model. The XY model is a two-dimensional spin model that exhibits a phase transition known as the Kosterlitz-Thouless transition. This transition is characterized by the unbinding of vortex-antivortex pairs, leading to a change in the behavior of the system. In this report, we will explore the theoretical basis of the XY model and the Kosterlitz-Thouless transition, as well as the methods used to study these phenomena and the results obtained from our simulations. Generally two algorithms are used to simulate the XY model: the Metropolis algorithm and the Wolff cluster algorithm. We will compare the results obtained from both algorithms and discuss their advantages and disadvantages in the context of simulating the XY model. In our simulations we will especially focus on the temperature dependence of magnetization and susceptibility, as well as determining the critical temperature of the Kosterlitz-Thouless transition. Finally, we will summarize our findings and discuss their implications for understanding phase transitions in two-dimensional systems. Additionally the methode of finite size scaling will be used to analyze the results and extract critical exponents associated with the Kosterlitz-Thouless transition. For higher dimensions the complexity of the XY model increases and the nature of the phase transition changes, which is why we will only shortly discuss it for higher dimensions.

II. THEORETICAL BASIS AND METHODS

The theoretical basis of the XY model and the Kosterlitz-Thouless transition is rooted in the study of phase transitions and critical phenomena in two-dimensional systems [1]. The XY model is a classical spin model that describes a system of spins on a two-dimensional lattice, where each spin can take any direction in the plane. The Hamiltonian of the XY model is given by:

$$H = -J \sum_{\langle i,j \rangle} \cos(\theta_i - \theta_j) \tag{1}$$

where J is the coupling constant (set to 1), $\langle i,j \rangle$ denotes nearest-neighbor pairs of spins, and θ_i is the angle of the spin at site i [2]. The Kosterlitz-Thouless transition is a topological phase transition that occurs in the XY model at a critical temperature $T_c \approx 1.12J$ [3, 4]. Below T_c, the system exhibits quasi-long-range order, characterized by a power-law decay of correlations, while above T_c, the system is in a disordered phase with exponential decay of correlations.

A. Markov Chain Monte Carlo Simulation

In order to give an overview of the methods used to simulate the XY model, we will briefly discuss the two main algorithms: the Metropolis algorithm and the Wolff cluster algorithm. Both algorithms are based on the principle of Markov Chain Monte Carlo (MCMC) simulation, which allows us to sample configurations of the system according to their Boltzmann weight [5]. The MCMC works by generating a sequence of configurations, where each configuration is generated from the previous one by applying a stochastic update rule. The Metropolis algorithm updates the spins one at a time, while the Wolff cluster algorithm updates clusters of spins simultaneously, which can lead to faster convergence and reduced autocorrelation times, especially near critical points [6]. In our simulations, we will compare the results obtained from both algorithms and discuss their advantages and disadvantages in the context of simulating the XY model [7].

B. Metropolis algorithm

Based on the MCMC method, the Metropolis algorithm is a widely used technique for simulating statistical sys-

tems [5]. In the context of the XY model, the Metropolis algorithm works as follows:

1. Initialize the system with a random configuration of spins.
2. For each Monte Carlo step, select a spin at random and propose a new angle for that spin by adding $\Delta\theta \sim U[0, 2\pi]$
3. Calculate the change in energy ΔE resulting from the proposed update.
4. Accept the proposed update with a probability given by the Metropolis criterion:

$$P_{\text{accept}} = \min\left(1, e^{-\Delta E/k_B T}\right) \tag{2}$$

5. If the update is accepted, update the configuration; otherwise, keep the current configuration.
6. Repeat the process for a large number of Monte Carlo steps to ensure convergence to equilibrium.

The `Metropolis()` method implements single-spin flip dynamics exactly as described above [2]. For each lattice site (x, y), it:

- Computes the local energy E_{now} using interactions with all four nearest neighbors
- Proposes $\theta_i' = \theta_i + \delta$ where $\delta \sim U[0, 2\pi)$ via `randomAngle()`
- Computes the new local energy E_{after}
- Accepts the move if $E_{\text{after}} < E_{\text{now}}$ or with probability $e^{-(E_{\text{after}} - E_{\text{now}})/T}$ via `random()`
- Updates $\theta_i \leftarrow \theta_i + \delta \mod 2\pi$

One full sweep iterates over all $N_x \times N_y$ sites sequentially [2].

C. Wolff algorithm

The Wolff algorithm is an efficient cluster update method that dramatically reduces critical slowing down near phase transitions [6]. Unlike single-spin flips, it builds and flips entire correlated clusters of spins:

1. Select a random seed spin s_0 and a random reflection angle $r \sim U[0, 2\pi)$.
2. Build a cluster by adding nearest neighbors with probability $p_{\text{add}} = 1 - \exp\left[\min(0, 2\Delta\theta\cos(\phi))\right]$, where ϕ is the relative angle after reflection.
3. Flip all spins in the cluster by $\theta_i \to 2r - \theta_i \mod 2\pi$.
4. Repeat until a target number of spins have been flipped (here: $N_x N_y$ total).

In `Wolff()`, clusters are grown using a stack-based algorithm with periodic boundary conditions, making it particularly effective for studying the XY model's BKT transition [2].

D. Finite Size Scaling (FSS)

Finite size scaling analyzes how observables behave as a function of system size $L = \sqrt{N}$ near criticality [8]. For the XY model near the Berezinskii-Kosterlitz-Thouless transition a kex observable is the susceptibility $\chi_L(T)$, which scales as:

$$\chi(L, t) \approx L^{\gamma/\nu}\, \hat{\chi}\left(L \exp\left(-b\, t^{-\nu}\right)\right) \tag{3}$$

, where $t = \frac{T_c - T}{T}$, η is the anomalous dimension, b is a non-universal constant, and θ is a critical exponent related to the correlation length [9].The simulations use grid sizes:

$$L = 8, 16, 32, 64, 128, 256$$

to extract T_c via data collapse [7].
The peak positions $T_{\max}(L)$ of the susceptibility were fitted using equation (4)

$$T(L) - T_c \propto \frac{1}{\log L} \tag{4}$$

to determine both T_c and the scaling factor.

E. Autocorrelation

Generally speaking, the autocorrelation function $C_X(t)$ of an observable X measures how correlated the values of X are at different times (or Monte Carlo steps) t. It is defined as:

$$C_X(t) \overset{\text{def}}{=} \left\langle (X_i - \mu)\,(X_{i+t} - \mu) \right\rangle \tag{5}$$

where μ is the mean of X and the average $\langle\cdot\rangle$ is taken over the Monte Carlo samples. The normalized autocorrelation function $\Gamma_X(t)$ is then given by:

$$\Gamma_X(t) \overset{\text{def}}{=} \frac{C_X(t)}{C_X(0)} \tag{6}$$

[10] It is a measure of how quickly the system "forgets" its past configurations.
$C_X(t)$ depends only on the time difference t. Typically, the autocovariance decays exponentially,

$$C_X(t) \propto \sum_{n \geq 0} A_n \exp\left(-\frac{|t|}{\tau_n}\right), \qquad \tau_0 > \tau_1 > \cdots \in \mathbb{R}^+. \tag{7}$$

For sufficiently large $|t|$ only the largest correlation time τ_0 survives. Hence, one defines the exponential autocorrelation time by

$$\tau_{\text{exp}} \overset{\text{def}}{=} \limsup_{t\to\infty} \frac{-t}{\ln\left|\Gamma_X(t)\right|}, \qquad \Gamma_X(t) = \frac{C_X(t)}{C_X(0)}. \tag{8}$$

[10]

III. ANALYSIS AND RESULTS

We will now investigate the response of the system to changes in various parameters. Throughout the investigation, physical constants such as K_B and J will be set to 1 for simplicity.

A. Temperature dependency

To quantify the temperature dependency of observables, the measurements have been performed with some considerations in mind. For the system to equilibrate, the first $N_{Burn} = 512$ steps have been discarded before recording any data. The data recording itself involved a total of $N_{Steps} = 512$ steps after reaching thermal equilibrium. Upon adjusting the temperature, the system was again given time to equilibrate again. The temperature has been set at equidistant points in the interval $T \in (0, 2]$, with $\Delta T = 0.02$. To perform sweep over the temperature grid, the system will start at the hot state, and gradually decrease in temperatre. The system will only be initialized to a random spin configuration when starting a new temperature sweep. It will not be reinitialize after each temperature step ΔT. This choice has been made deriberately, due to the large correlation time of specifically the Metropolis algorithm at lower temperatures, which would require larger N_{Burn} to reach thermal equilibrium, especially when starting from a hot state. It would mean that T_i and T_{i+1} are slightly correlated, however by choosing N_{Burn} appropriately, this correlation can be kept small, as seen in later sections. The entirety of the measurement comprises a total 20 temperature sweeps for all grid sizes in $L = 8, 16, 32, 64, 128, 256$.

Fig. 1 displays the temperature dependency of all four observables performed using the Metropolis algorithm. When all spins are aligned at $T = 0$, $\langle E\rangle$ and $\langle M\rangle$ reach their expected value of -2 and 1 respectively. Signs of a phase transition can be seen around temperatures where C_v peaks and *chi* diverges, and $\langle E\rangle$ and $\langle M\rangle$ show (strong) slope changes. For higher lattice sizes, these features become even more pronounced, which is due to the finite size effects. At higher temperatures, the spin alignment is almost completely broken, indicated by $\langle E\rangle$ and $\langle M\rangle$ approching 0.

Fig. 2 displays the temperature dependency for the same

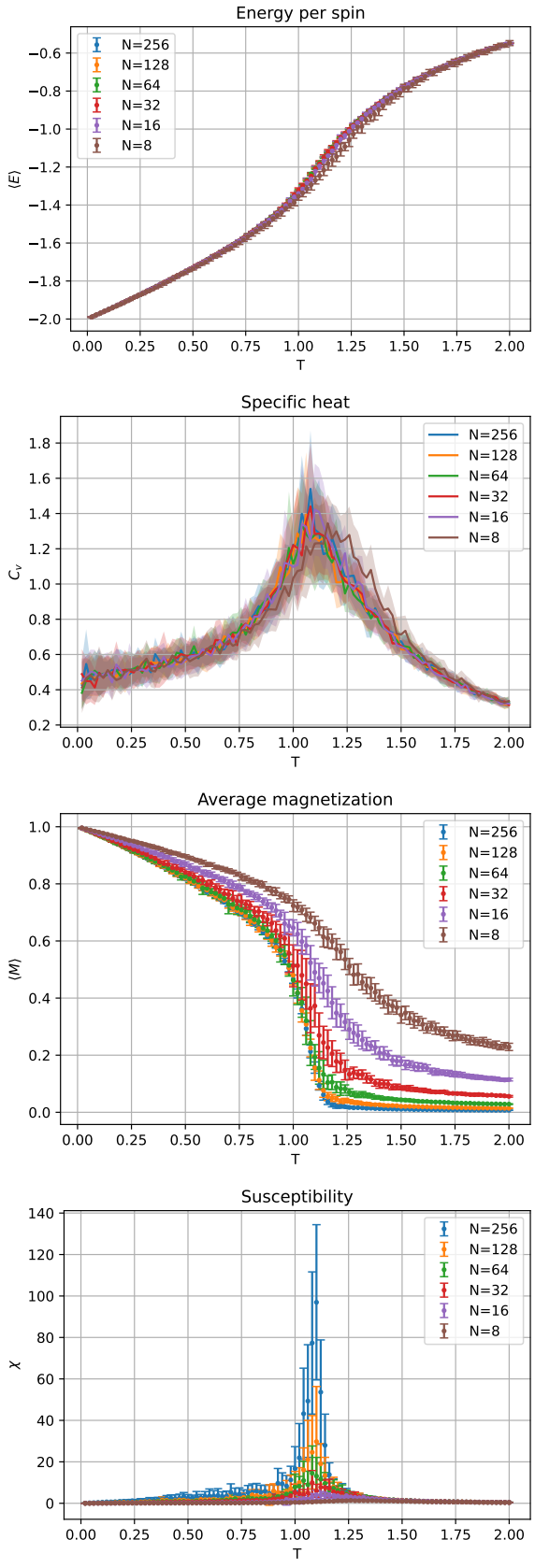

FIG. 1. Temperature dependency of the observables using the Metropolis algorithm

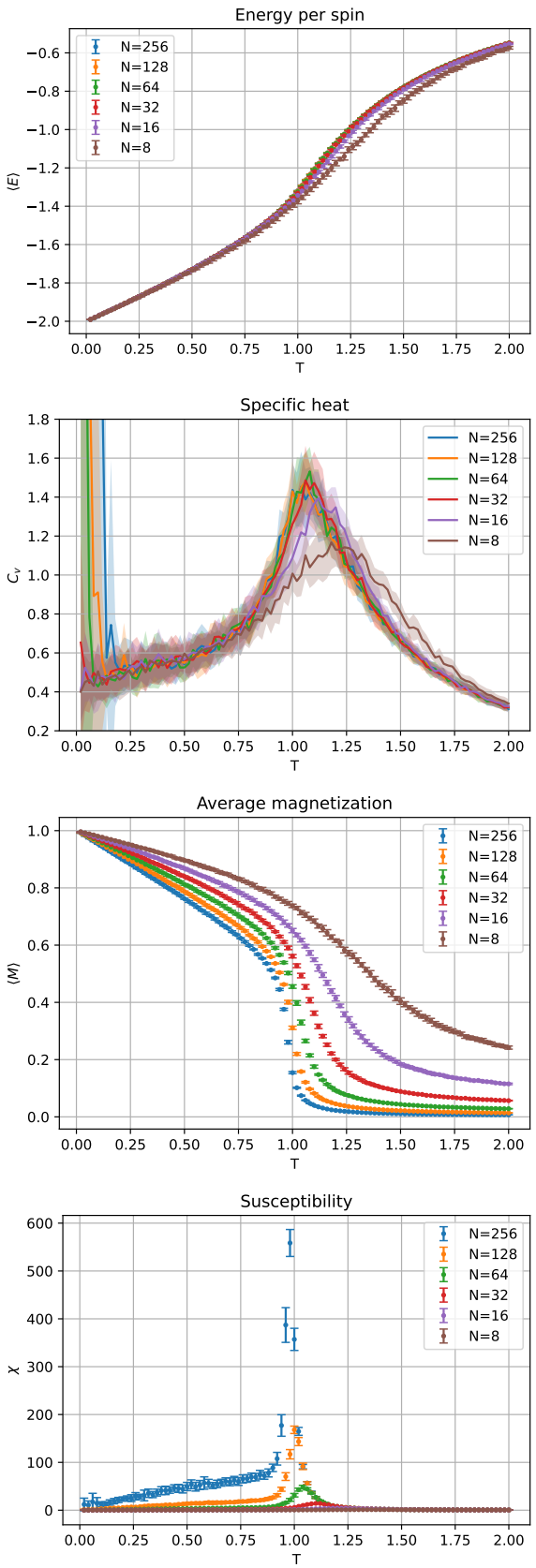

FIG. 2. Temperature dependency of the observables using the Wolff algorithm

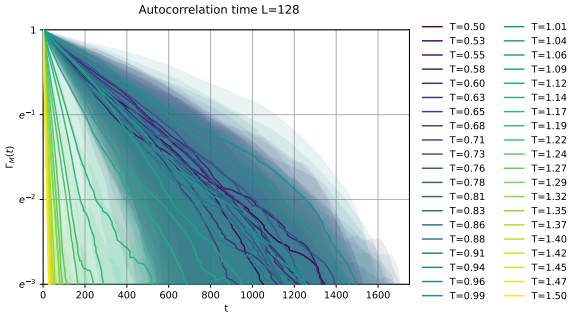

FIG. 3. Autocorrelation values as a function of time steps t for the Metropolis algorithm at $L = 128$.

observables performed using the Wolff algorithm. The same behavior can be can be seen for this case as well. The main difference lies in the reduced variance of the measurements, which is a consequence of the reduced correlation time of the Wolff algorithm. Furthermore, the peaks for the susceptibility are more pronounced and shifted more towards the theoretical values for T_c. The single spin updates of the Metropolis algorithm are heavily affected by critical slowing down, resulting in a larger correlation time and thus higher variance of the measurements.

B. Autocorrelation time

We have already seen the effects of autocorrelation times on the measurements in the previous subsection, and now we want to quantify them. The autocorrelation function is given by eq. (6), which is used to measure the time steps needed for two spin configurations to become statistically independent, according to eq. (8). In order to accurately compare the two algorithm, we defined a time step for an algorithm as the attempt to flip all the spins on the lattice. At a given temperature, the system will be initialized to a cold state for quicker equilibration in the burn-in phase with $N_{Burn} = 500$. After reaching thermal equilibrium, a total of $N_{Meas} = 5000$ samples are taken to make sure that the autocorrelation time τ is well within that range, and for a more accurate estimation of the mean of the observable. The absolute value of the magnetization $|M|$ is chosen as the observable. Then, the autocorrelation function is calculated for all time steps until $\Gamma_M(t) < e^{-3}$, after which there was mostly just noise. Such a measurement is then repeated $N_{Rep} = 20$ times. This will be repeated for all temperatures in $T \in [0.5, 1.5]$ with 40 temperature points inside, and all lattice sizes in $L = 8, 16, 32, 64, 128$.

Fig. 3 visualizes the autocorrelation values as a function of time steps for different temperatures when using the Metropolis algorithm. Apparent is the separation between lower and higher temperatures, where lower temperatures show a much slower decay than higher temper-

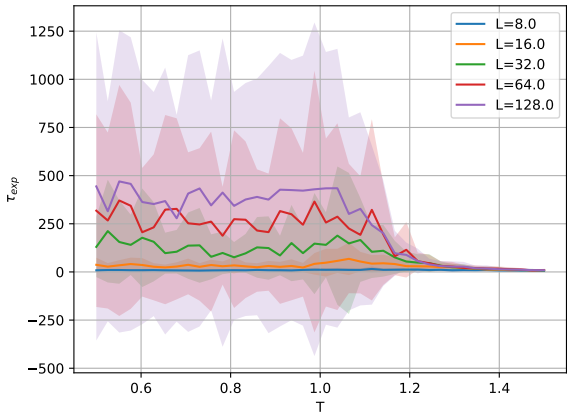

FIG. 4. Autocorrelation times as a function of temperature for the Metropolis algorithm for different lattice sizes.

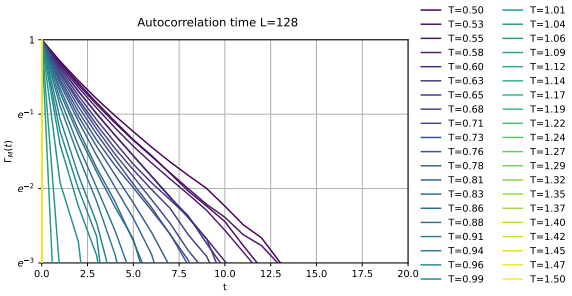

FIG. 5. Autocorrelation values as a function of time steps t for the Wolff algorithm at $L = 128$.

atures. In Fig. 4 the autocorrelation time has been extracted, which highlights this effect. The autocorrelation times seem to stabilize around a value for $T < T_c$, and gradually decrease for $T > T_c$. For lower temperatures, single spin updates struggle to flip large correlated clusters. Only along the border of such clusters lies the highest probability for a proposed spin flip to be accepted, so the time required for the whole cluster to change its spin orientation grows, as the cluster size grows.

Fig. 5 and Fig. 6 now display the autocorrelation values and times for the Wolff algorithm. The same general behavior can be seen here as well; higher autocorrelation times for lower temperatures. But unlike the case of the Metropolis algorithm, the autocorrelation times are significantly lower, up to 2 orders of magnitude. This demonstrates that cluster updates are more efficient at producing uncorrelated spin configurations.

C. Critical Temperature

In order to estimate the critical temperature, we have to deal with the finite size effects, which cause the peaks of the susceptibility to be shifted for different lattice sizes.

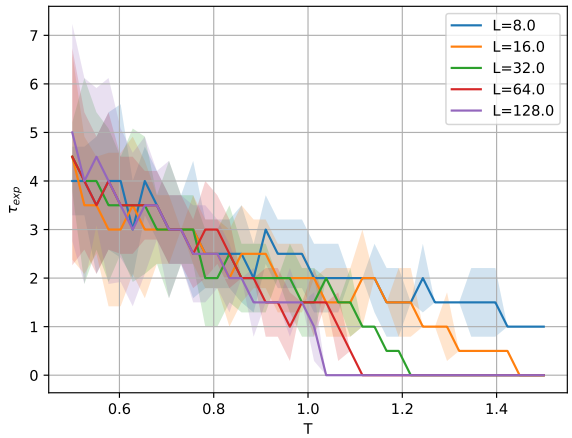

FIG. 6. Autocorrelation times as a function of temperature for the Wolff algorithm for different lattice sizes.

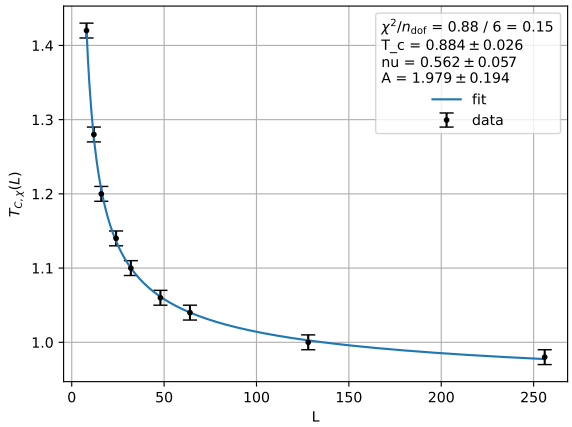

FIG. 7. Plottet are the temperatures at which χ peaks for various lattice sizes. Fittet through is a line according to eq. (4).

By determining the temperature at which the peaks arise, we can extrapolate the critical temperature for an infinite lattice size according to eq. (4). Performing a line fit through the peak temperatures will then give us an estimate for T_c and the critical exponent ν.

Fig. 7 displays the peak temperatures $T_{C,\chi}(L)$ with a line fit using $T_c(L) = T_c(\infty) + \mathrm{A} \cdot \ln(L)^{-1/\nu}$. The fittet parameters are found to be $T_c(\infty) = 0.884 \pm 0.026$ and $\nu = 0.562 \pm 0.057$. Within the standard deviation, the estimated critical temperature is consistent with the theoretical value of $T_c = 0.89294$ [11]. The critical exponent for the correlation length is estimated to be $\nu = 0.562 \pm 0.057$, which is close but outside of the range for the theoretical value of $\nu = \frac{1}{2}$ [9, 11]. Adding more data points for larger lattice sizes could potentially improve the estimation of ν and the reduced chi-squared value of the fit of $\chi^2/n_{\mathrm{dof}} = 0.15$.

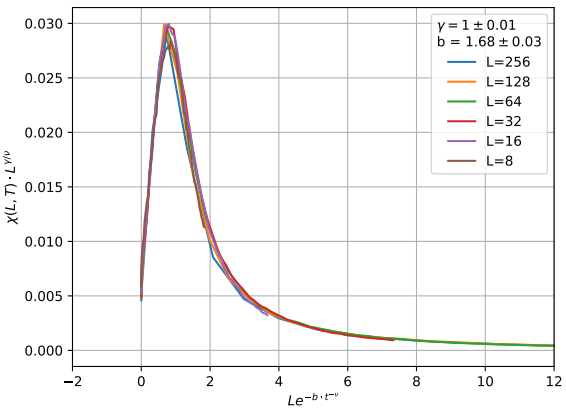

FIG. 8. Rescaled values of the susceptibility $\chi(L,t) \cdot L^{-\frac{\gamma}{\nu}}$ plotted against $L \cdot \exp(-b \cdot t^{-\nu})$ for all lattice sizes. T_c and ν have been taken from the previous section, while γ and b are estimated by eye to achieve the best collapse.

D. Data Collapse

Using the estimated values for T_c and ν, we can perform a data collapse technique for the susceptibility, by scaling the coordinates according to eq. (3). By plotting the the scaled values of the measured susceptibility $\chi(L,t) \cdot L^{-\frac{\gamma}{\nu}}$ against $L \cdot \exp(-b \cdot t^{-\nu})$, the datapoints for all lattice sizes should collapse onto a single curve, given appropriate values for the parameters are provided.

Fig. 8 illustrates the data collapse using visually estimated parameters. Choosing $\gamma = 1 \pm 0.01$ and $b = 1.68 \pm 0.03$, the data collapse seems visually acceptable. This estimation for the critical exponent of the susceptibility $\gamma = 1 \pm 0.01$ is consistent with the theoretical ratio of $\gamma/\nu = 2 - \eta$ [4], with $\eta = 1/4$, so that $\gamma/\nu = 1.75$, which is within the confidence interval.
It is possible to estimate all four parameters independently of the curve fit from the last subsection, however it turned out to be very difficult to balance them out for an acceptable data collapse simply by visual judgement. We also attempted to quantify the data collapse by measuring the squared distances of the peaks on the scaled χ values, and minimizing this quantity using the `migrad()` function in the `iMinuit` Python library. However, this approach collapsed the data onto either a single point or a single straight line, yielding unreasonable values for the parameters.

IV. DISCUSSION

In this section the results of the Metropolis algorithm and Wolff algorithm are compared and discussed. The results of the Metropolis algorithm are shown in Figure 1 and the results of the Wolff algorithm are shown in Figure 2. Noticeable is the reduced variance of the measurements for the Wolff algorithm, which is a consequence of the reduced correlation time. Furthermore, the peaks for the susceptibility are more pronounced and shifted more towards the theoretical values for T_c. The single spin updates of the Metropolis algorithm are heavily affected by critical slowing down, resulting in a larger correlation time and thus higher variance of the measurements. This is further confirmed by Figure 4, where we see that the autocorrelation time τ increases significantly around T_c for all lattice sizes, especially for larger lattice sizes. This is a clear indication of critical slowing down in the Metropolis algorithm, which is not present in the Wolff algorithm due to its cluster updates. Generally the more measurements are taken, the more accurate the results will be, however this also increases the computational time. The Wolff algorithm allows for more measurements to be taken in a given time frame due to its reduced correlation time. Finer and wider temperature grid, especially around T_c would enable us to better capture the behavior of the system around the critical temperature, and thus more accurately determine T_c and the critical exponents. You could further improve the accuracy by estimating τ between every measurement, and only record measurements that are statistically independent, which would further reduce the variance of the measurements. However this would also increase the computational time, especially for the Metropolis algorithm, where τ is significantly larger around T_c. The Wolff algorithm would benefit less from this improvement, as its correlation time is already significantly reduced compared to the Metropolis algorithm.

Appendix A: Appendix

[1] J. M. Kosterlitz and D. J. Thouless, J. Phys. C: Solid State Phys. **6**, 1181 (1973).
[2] K. Helders and C. Hübbel, Xymodel.hpp (2026), implementation of XY model.
[3] M. Hasenbusch, Phys. Rev. B **82**, 174433 (2010).
[4] J. M. Kosterlitz, J. Phys. C: Solid State Phys. **7**, 1046 (1974).
[5] N. Metropolis and S. Ulam, Journal of the American Statistical Association **44**, 335 (1949).
[6] U. Wolff, Phys. Rev. Lett. **62**, 361 (1989).
[7] K. Helders and C. Hübbel, main_multithreading.cpp (2026), multithreaded simulation driver.
[8] V. Privman, Physica A: Statistical Mechanics and its Applications **177**, 241 (1991).
[9] N. Sale, J. Giansiracusa, and B. Lucini, Quantitative analysis of phase transitions in 2d xy models using persistent homology (2021).
[10] M. Petschlies and C. Urbach, Computational physics (lecture script) (2026), bonn University; version dated 19 January 2026; CC BY-NC-SA.
[11] M. Hasenbusch, Journal of Physics A Mathematical General **38**, 5869 (2005), arXiv:cond-mat/0502556 [cond-mat.stat-mech].