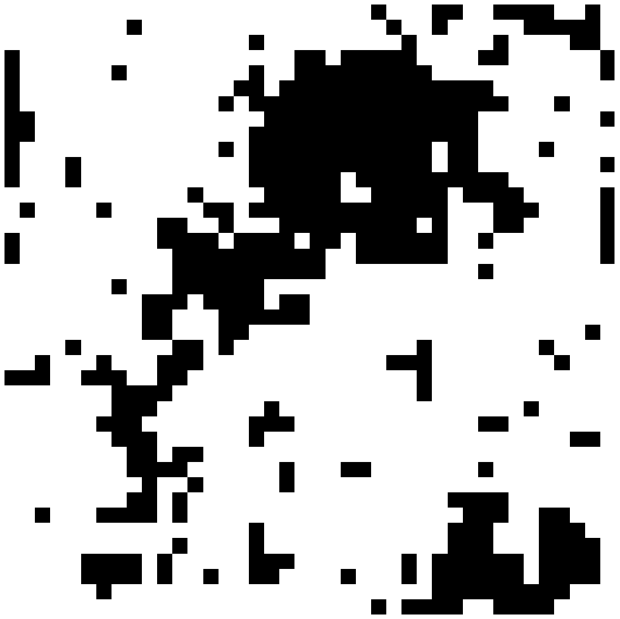
Physics or Gambling? A Monte Carlo Simulation of the Ising Model

Adrian Gundersen,[*] Casper André Johnsen,[†] and Victor Berge Johansen[‡]
(Dated: November 20, 2025)

We have studied qualitative behavior during phase transitions for a ferromagnet by simulating a two-dimensional Ising model with Markov Chain Monte Carlo methods (MCMC). Utilizing the Metropolis–Hastings algorithm, we analyze properties of the system. We also calculate uncertainties using error propagation and discuss autocorrelation between cycles. Firstly, we verified against analytical results that the Metropolis algorithm is a useful algorithm for the Ising model. Further, we studied the required burn-in time and found that $\approx 10{,}000$ cycles should be sufficient. We then parallelized the code over different walkers to get lower uncertainties and to speed up the code. For threads ≤ 6 the speedup was almost ideal, but the gains diminished for higher threads. With parallelized code, we found the probability distributions of the energy per spin ϵ for $T = 1.0\,J/k_B$ and $T = 2.4\,J/k_B$. For $p(\epsilon; T = 2.4\,J/k_B)$ it seemed to follow a Gaussian probability distribution, but for $T = 1.0\,J/k_B$ there were only two noticeable peaks at the lowest energies. We then looked at qualitative properties for different temperatures to find the critical temperature T_c and showed it was proportional to $1/L$. Taking a linear regression of T_c against $1/L$ and letting $L \to \infty$, we found that $T_c(L = \infty) = 2.26917 \pm 0.00014\,J/k_B$, giving a $\sim 6.75 \cdot 10^{-6}$ relative error against Lars Onsager's analytical result. All results can be reproduced by running the code at our GitHub repository[a] under `Project4/`.



Ising model of size 40×40.

I. INTRODUCTION

Phase transitions and critical points are central topics in both statistical mechanics and thermodynamics. At the critical point small changes in external parameters can give large macroscopic changes of a system. This behavior can be hard to study, especially on a microscopic level. However, many systems exhibit similar qualitative behavior around the critical temperature. So by studying one system, we can transfer the insight to multiple systems.

In our work we simulate a two-dimensional ferromagnetic Ising model on a square lattice. The model consists of spins $s_i = \pm 1$ on lattice sites, where the energy and magnetization only depend on its neighbors. Despite the simplicity, we want to show that it can give good qualitative insight into complex systems like ferromagnets. It is also often used in describing voter models, lattice gases, neuroscience and artificial neural networks.[1]

Although the Ising model is a simplified picture of reality, analytical solutions are hard for large lattices. We must therefore rely on numerical approximations. In this paper we will utilize Markov Chain Monte Carlo

[*] adriangg@uio.no
[†] casjoh@uio.no
[‡] victorbj@uio.no
[a] https://github.uio.no/adriangg/FYS3150

(MCMC) methods to simulate the Ising model. However, we must verify that our method is correct to be able to use the results. So we compare the MCMC algorithm to known analytical solutions for $L = 2$ and study convergence and burn-in. Further, we also give estimated values for uncertainties to see if our results are reasonable.

Although MCMC simulations are easier than obtaining analytical solutions for big lattices, it is still both time and computationally expensive. To make it easier, we optimize and parallelize the code. By parallelizing over different walkers, using OpenMP, we decrease both runtime and uncertainty. To reduce heavy computations we also implement efficient boundary conditions, data structures and precompute exponential factors and discuss the impact.

We will in this report lead off with an explanation of the physics, the model and the approach to simulating the system in Section II. Here we go through expressions for observables and critical points as well as give an overview of our implementation of the Metropolis algorithm. Having run the simulations, we present and discuss the results in Section III before concluding in IV. Here we also discuss what could be improved to better our results and build a better framework for Ising model analysis. To find the code used to produce the results see our GitHub repository [a] under `Project4/`. The `README.md` explains the code and what parameters we used. We also have included `data/tsweep_finale.json` which is our output file from all the temperature sweeps including core parameters and observables.

II. METHODS

A. Physical Model

We will be looking at a two-dimensional Ising model.

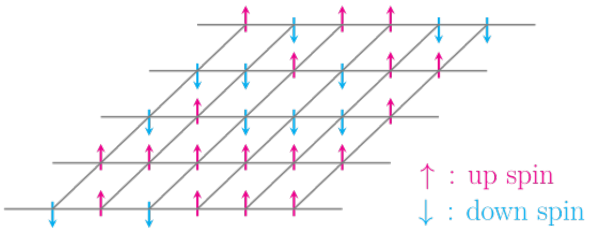


FIG. 1: Illustration of the 2D Ising model.[2]

The model is a lattice with multiple spins, as depicted in Figure 1. In our case the lattice will always be a square of length L. We define the lattice size $N \equiv L^2$ as the number of spins and construct a spin configuration

$$\mathbf{s} = (s_1, s_2, \dots, s_N).$$

This can also be written in matrix form as:

$$\mathbf{s} = \begin{pmatrix} s_{11} & s_{12} & \cdots & s_{1L} \\ s_{21} & s_{22} & \cdots & s_{2L} \\ \vdots & \vdots & \ddots & \vdots \\ s_{L1} & s_{L2} & \cdots & s_{LL} \end{pmatrix}.$$

Each spin can attain values up or down: $s_i \in \{-1, +1\}$. However, the spins can flip and interact with their neighbors. The spins tend to states that minimizes energy. The system is also in contact with a thermal bath which we consider to have a constant temperature T.

1. *Observables*

We will be looking at multiple observables of the system by using known expressions. For full derivations, see [3].

The total energy of the system is given as

$$E(\mathbf{s}) = -J \sum_{\langle ij \rangle}^{N} s_i s_j,$$

where $\langle ij \rangle$ is the sum over all neighboring pairs of spins, without double-counting, and J is the coupling constant [4]. To ensure all spins have exactly four neighbors we use periodic boundary conditions. The total magnetization of the system is given as:

$$M(\mathbf{s}) = \sum_{i=1}^{N} s_i.$$

We also define the energy per spin ϵ and magnetization per spin m:

$$\epsilon(\mathbf{s}) \equiv \frac{E(\mathbf{s})}{N} \quad m(\mathbf{s}) \equiv \frac{M(\mathbf{s})}{N}.$$

For a given system temperature T, we also have the Boltzmann probability distribution

$$p(\mathbf{s}) \equiv p(\mathbf{s}; T) = \frac{1}{Z} e^{-\beta E(\mathbf{s})}. \tag{1}$$

Describing the probability of the system being in state $\mathbf{s}$. Here $\beta = 1/k_B T$, where k_B is the Boltzmann constant, and Z is the partition function

$$Z = \sum_{\text{all states } \mathbf{s}} e^{-\beta E(\mathbf{s})}.$$

For an observable $X_\mathbf{s}$ this makes the expectation value

$$\langle X \rangle = \sum_{\mathbf{s}} p(\mathbf{s}) X_\mathbf{s}.$$

Further, we have the specific heat capacity c_V and susceptibility χ_s given as

$$c_V = \frac{C_V}{N} = \frac{1}{N k_B T^2} \text{Var}(E) = N \frac{1}{k_B T^2} \Big(\langle \epsilon^2 \rangle - \langle \epsilon \rangle^2 \Big),$$

$$\chi_\text{s} = \frac{\chi}{N} = \frac{1}{N k_B T} \Big(\langle M^2 \rangle - \langle |M| \rangle^2 \Big) = \frac{N}{k_B T} \Big(\langle m^2 \rangle - \langle |m| \rangle^2 \Big).$$

Where we have utilized a trick, letting $M \to |M|$ as $\langle M \rangle \approx 0$ for finite systems. This makes it easier to compare results. [4] The heat capacity and susceptibility derivations can be found in Appendix A 2.

2. *Spin flips*

If we consider a single spin flip, we want to find the values the change in energy $\Delta E = E' - E_0$ can take for a general $L \times L$ lattice. Flipping the sign of one spin s_i can only change the bonds to its four neighbors s_j. So the change per bond is $2Js_is_j$, giving $\Delta E = \sum_j 2Js_js_i$. Since $\sum_j s_j \in \{-4, -2, 0, 2, 4\}$, the allowed energy changes are

$$\Delta E \in \{-8J,\ -4J,\ 0,\ 4J,\ 8J\}.$$

As the spins tend to minimize energy, if $\Delta E \leq 0$ the spin will always flip. However, there is also a probability for the spin to flip even when it will increase the energy as described by the Boltzmann probability distribution in Equation (1).

3. *Units*

To ease our calculations we will use energy unit $[E] = J$, where J is the coupling constant. We also let $J = 1$ in reduced units.

TABLE I: Reduced units used for Ising model. We set $J \equiv 1$.

Quantity	**Symbol**	**Unit (natural)**
Lattice size	L, N	1
Energy	E	J
Temperature	T	J/k_B
Spin	s	1
Magnetization	M	1
Heat capacity	C_V	k_B
Susceptibility	χ	$1/J$

In Table I we have listed the derived units of all quantities we will look at.

4. *Phase transitions, critical temperature & linear fit*

A phase transition is a big change in the macroscopic state, for instance water to ice. In the Ising model this occurs when we vary temperature T. We can then view the physical properties and behavior with respect to the temperature and how they change during the phase transition. near a phase transition small changes in temperature lead to large change in internal energy making its heat capacity C_V diverge. At the critical temperature T_c the spontaneous magnetization $\langle |m| \rangle$ vanishes, and the system becomes very sensitive to an external magnetic field. This enhanced response is described by the magnetic susceptibility χ_s.

When a ferromagnetic material reaches T_c it spontaneously looses its magnetization and becomes a paramagnet. This temperature is also known as the Curie temperature T_C.[3] As the system approaches T_C and $\langle |m| \rangle \to 0$, the susceptibility χ diverge. This is formally stated in Curie-Weiss law:

$$\chi = \frac{C}{T - T_C}, \qquad T > T_C,$$

where C is a material specific constant. The law fails to describe the behavior close to the critical point, and for this model scales wrong thus we use an altered version which describes χ_s behavior for an infinite system and approximate its behavior around the critical temperature. [5]

$$\chi_s \sim |T - T_C|^{-\gamma}. \tag{2}$$

Where $\gamma = 7/4$ and $\nu = 1$ for the 2D Ising model.[4] When looking at the maxima of χ_s it has a dependence on L given as:

$$\chi_\text{s} \propto L^{\gamma/\nu}. \tag{3}$$

We also expect c_V to diverge as the system needs a lot of energy to change phase, this relation is given as

$$c_V \propto L^{\alpha/\nu},$$

where $\alpha = 0$ corresponds to a marginal case, where the specific heat capacity c_V at the critical point grows logarithmically with system size,

$$c_V \propto \ln L.\text{[4]} \tag{4}$$

at T_C the magnetization $\langle |m| \rangle$ drops according to:

$$\langle |m| \rangle \propto L^{-\beta/\nu},$$

where $\beta = 1/8$. These relations imply that χ_s and c_V diverge as $L \to \infty$ and $\langle |m| \rangle \to 0$.

For simplicity and consistency we denote the Curie temperature and the critical temperature as T_c to stay consistent as they are equal.

In 1944 Lars Onsager found that the critical temperature for an infinite ($L = \infty$) 2D Ising model is given as:

$$T_c(L = \infty) = \frac{2}{\ln\left(1 + \sqrt{2}\right)}\, J/k_B \approx 2.269185311421\, J/k_B.\text{[6]}$$

In a finite Ising model we define the critical temperature $T_c(L)$. To find $T_c(L)$ as a function of lattice size L numerically, we use the maximum value of the specific heat capacity c_V and susceptibility χ_s. For increasing values of L we calculate

$$T_c^{(c_V)}(L) = \arg\max_T \left[c_V(L;T)\right],$$
$$T_c^{(\chi_\mathrm{s})}(L) = \arg\max_T \left[\chi_\mathrm{s}(L;T)\right].$$

However, due to discrete datapoints and statistical uncertainty, this is not the actual maximum. Thus, we do a second order polynomial fit around the maximum given as

$$f(T) = aT^2 + bT + c$$

to both $c_V(T;L)$ or $\chi_\mathrm{s}(T;L)$. The extremum of f is then at

$$\hat{T}_c = -\frac{b}{2a},$$

with uncertainty

$$\delta(\hat{T}_c) = \sqrt{\left(\frac{\partial T}{\partial a}\right)^2 \mathrm{Var}(a) + \left(\frac{\partial T}{\partial b}\right)^2 \mathrm{Var}(b) + 2\frac{\partial T}{\partial a}\frac{\partial T}{\partial b}\,\mathrm{Cov}(a,b)},$$

where, $\mathrm{Var}(a)$, $\mathrm{Var}(b)$ and $\mathrm{Cov}(a,b)$ comes from `numpy.polyfit`.

This gives us our respective critical points $\hat{T}_c^{(c_V)}(L)$ and $\hat{T}_c^{(\chi_\mathrm{s})}(L)$. By taking the mean, we get the critical temperature at L as

$$\hat{T}_c(L) = \frac{\hat{T}_c^{(c_V)}(L) + \hat{T}_c^{(\chi_\mathrm{s})}(L)}{2}. \quad (5)$$

This gives us estimated critical temperatures $\hat{T}_c(L)$ as a function of lattice length L. To find the critical temperature at infinity we use the relation

$$\hat{T}_c(L) - T_c(L=\infty) = aL^{-1}, [4] \quad (6)$$

with proportionality constant a. So we do a linear regression

$$T_c(L) = b + a\frac{1}{L},$$

with slope a. As $L \to \infty$, the critical temperature for an infinite lattice becomes the intercept $b = T_c(L=\infty)$.

In general for a linear relation $y = ax + b$, we use $\frac{\partial b}{\partial a} = -x$ and $\frac{\partial b}{\partial \bar{y}} = 1$ to find the uncertainty as

$$\delta(T_c(L=\infty))^2 = \mathrm{Var}(\bar{y}) + \bar{x}^2\,\mathrm{Var}(a).$$

Where the variances are

$$\mathrm{Var}(\bar{y}) = \frac{\sigma_y^2}{n_\mathrm{pts}} \quad , \quad \mathrm{Var}(a) = \frac{\sigma_y^2}{\sum_{n_\mathrm{pts}}(x_i - \bar{x})^2},$$

and n_pts is the number of data points in the linear regression. The uncertainty in y is given by

$$\sigma_y^2 = \frac{1}{2}\left[(\delta\hat{T}_c^{(c_V)})^2 + (\delta\hat{T}_c^{(\chi_\mathrm{s})})^2\right].$$

Putting it all together and rewriting, the uncertainty of the intercept in Equation 6 becomes

$$\delta(T_c(L=\infty)) = \sigma_y\sqrt{\frac{\sum_{n_\mathrm{pts}} x_i^2}{n_\mathrm{pts}\sum_{n_\mathrm{pts}}(x_i - \bar{x})^2}}.$$

It is worth noting that our T_c for a finite model is in fact not a phase shift as all of our observables will be analytic for T. For a real phase shift c_V, χ_s will diverge. It is however a critical point analogue to a phase transfer so we will denote it as such.

5. 2×2 *example*

To compare with analytical results, we consider a 2×2 lattice with periodic boundary conditions.

TABLE II: All macrostates for 2×2 Ising model.

# $s_i = +1$	**Energy** E	**Magnetization** M	**Degeneracy** g
4	$-8J$	4	1
3	0	2	4
2 (A)	$8J$	0	2
2 (B)	0	0	4
1	0	-2	4
0	$-8J$	-4	1

In Table II, 2 spins $s_i = +1$ can come from either checkerboard pattern (A) or stripe pattern (B).

From our probability distribution $p(\mathbf{s})$ in Equation (1), we can derive some useful expressions for the 2×2 Ising model $N = 4$.

$$Z = 12 + 4\cosh(8\beta J),$$
$$\langle\epsilon\rangle = -\frac{8J}{Z}\sinh(8\beta J),$$
$$\langle\epsilon^2\rangle = \frac{16J^2}{Z}\cosh(8\beta J),$$
$$\langle|m|\rangle = \frac{2}{Z}(e^{+8\beta J} + 2),$$
$$\langle m^2\rangle = \frac{2}{Z}(e^{+8\beta J} + 1),$$
$$c_V = \frac{4}{k_B T^2}\left(\langle\epsilon^2\rangle - \langle\epsilon\rangle^2\right),$$
$$\chi_\mathrm{s} = \frac{4}{k_B T}\left(\langle m^2\rangle - \langle|m|\rangle^2\right).$$

Derivations can be found in Appendix A 1.

B. Simulation & Monte Carlo

1. *Markov chains*

A Markov chain is a stochastic process where the next state depends solely on the current. So for a state space

$S = \{X_i\}$, we have the identity

$$P(X_i \to X' \mid X_i, X_{i-1}, \dots, X_1) = P(X_i \to X' \mid X_i).$$

In our case a state X is the configuration of spins in the $L \times L$ lattice, $\mathbf{s}$. It is also normal to define walkers that start in a given state and are developed by the Markov chain.

We let $\pi^{(n)}$ denote the row vector of probabilities after n Monte Carlo steps,

$$\pi^{(n)} = \left[\pi_1^{(n)}, \pi_2^{(n)}, \dots, \pi_N^{(n)}\right],$$

where $\pi_k^{(n)}$ is the probability that the chain is in state X_k at step n, and N is the number of possible spin configurations. The transition probabilities between states can be collected in a transition matrix $A \in \mathbb{R}^{N\times N}$ with elements

$$A_{ij} = P(X_i \to X_j \text{ in one step}).$$

The Markov chain then evolves according to

$$\pi^{(n+1)} = \pi^{(n)}\mathbf{A}.$$

Further, if

$$\pi\mathbf{A} = \pi,$$

π is a stationary distribution meaning that it is unchanged by the transition. A Markov Chain is also ergodic if it converges to the same stationary distribution, regardless of initial distribution. We will also introduce multiple walkers. We will run multiple walkers in parallel. Each walker will then be an independent Markov chain with the same transition matrix and stationary distribution. Observables will then converge to the same value. [7]

2. *Monte Carlo & law of large numbers*

For larger lattices, the number of possible states grows exponentially and it becomes practically impossible to compute an exact partition function Z. This makes it hard to find an accurate probability distribution and expected observable values.

Monte Carlo methods, named after the Monte Carlo Casino in Monaco makes this easier. We only propose local spin flips and not the entire model in each Markov step. We therefore construct a local update rule deciding whether or not the spin should flip. This is justified by the law of large numbers. If the Markov chain is ergodic and has a stationary distribution (like the Boltzmann distribution), then the sample average of an observable will converge towards the expectation value. The histogram will also approach the probability distribution of the observable.

3. *Uncertainty of Monte Carlo*

Performing integration using the Monte Carlo simulation gives an uncertainty of order $\mathcal{O}\left(1/\sqrt{n}\right)$ for n independent samples. To derive exact expressions we recall that a random variable X can be estimated using the simple variance

$$\mathrm{Var}(X) \approx s_X^2 = \frac{1}{(n-1)}\sum_{i=1}^{n}(X_i - \bar{X})^2.$$

From this we get the standard deviation $\sigma_X \approx s_X$. Computationally it is simpler to use the known relation

$$s_X^2 \approx \langle X^2\rangle - \langle X\rangle^2.$$

The uncertainty in $\bar{X}$, then becomes the standard error

$$\delta(\hat{X}) = \sqrt{\frac{\mathrm{Var}(X)}{n}} = \frac{\sigma_X}{\sqrt{n}}.$$

So the expectation value of X becomes

$$\langle X\rangle \approx \bar{X} \pm \frac{\sigma_X}{\sqrt{n}}.$$

For an observable that is a function of several means

$$F = f\big(\mu_1, \dots, \mu_p\big), \quad \mu_j \equiv \langle X_j\rangle,$$

we utilize the first order propagation of uncertainty giving

$$\mathrm{Var}(F) \approx \nabla f^\top\, \mathrm{Cov}(\boldsymbol{\mu})\, \nabla f,$$

with the uncertainty in F as

$$\delta(F) = \sqrt{\mathrm{Var(F)}},$$

where we do not divide by $\sqrt{n}$ as F is a function of means μ_i.

This can be used to find the uncertainty in the data points for c_V and χ_s for fixed temperatures. We introduce notation $e_n = \mathrm{Var}(\epsilon^n)$ and $v_n = \mathrm{Var}(|m|^n)$. Then we calculate the error as

$$\delta(c_V) = \left(\frac{N}{k_BT^2}\right)\sqrt{\frac{e_2}{n} + 4\langle\epsilon\rangle^2\frac{e_1}{n} - 4\langle\epsilon\rangle\frac{\mathrm{Cov}(\epsilon^2,\epsilon)}{n}},$$
$$\delta(\chi_\mathrm{s}) = \left(\frac{N}{k_BT}\right)\sqrt{\frac{v_2}{n} + 4\langle|m|\rangle^2\frac{v_1}{n} - 4\langle|m|\rangle\frac{\mathrm{Cov}(m^2,|m|)}{n}}.$$

The full derivation can be found in Appendix A 3.

Introducing W independent walkers reduces the uncertainty to

$$\delta(c_V) = \frac{1}{\sqrt{Wn}}\left(\frac{N}{k_BT^2}\right)\sqrt{e_2 + 4\langle\epsilon\rangle^2 e_1 - 4\langle\epsilon\rangle\,\mathrm{Cov}(\epsilon^2,\epsilon)},$$
$$\delta(\chi_\mathrm{s}) = \frac{1}{\sqrt{Wn}}\left(\frac{N}{k_BT}\right)\sqrt{v_2 + 4\langle|m|\rangle^2 v_1 - 4\langle|m|\rangle\mathrm{Cov}(m^2,|m|)},$$

as the walkers are independent. The expressions tell us that for a larger grids we can expect more noise and a higher uncertainty in the data. However, assuming that the noise is normally distributed around the true value this will not affect our values for $\hat{T}_c$ as this uncertainty comes from how good the polynomial fit is as described in Section II A 4.

We have assumed independent samples, which is a vast oversimplification. In reality we should look at the effective samples n_{eff}. We would then have to compute autocorrelation functions which would require us to store much larger datasets as well. This is however beyond the scope of our research, but should be considered in future work.[8]

4. *Burn-in*

As the initial state has a probability of being an unlikely state, it can skew the distribution towards a value that is unlikely to be attained. We will therefore let the walkers stabilize before we use the measurements. To study this we will look at the energy per spin ϵ and see how many cycles it takes before it stabilizes.

Too many burn-in cycles will not affect the results negatively, but will however take more time. Still it is better to have too many burn-in cycles than too few.

5. *Efficient boundary conditions and spin array*

When calculating properties of the system throughout the matrix, we want to avoid multiple if-statements around the boundaries to reduce computation speed. For this approach we use the modulus of the position, for position i we want to go left and right. We denote mod as the modulus operator and get:

$$i_{\text{right}} = (i+1) \bmod L,$$
$$i_{\text{left}} = (i+L-1) \bmod L.$$

From position j up and down is found by:

$$j_{\text{down}} = (j+1) \bmod L,$$
$$j_{\text{up}} = (j+L-1) \bmod L.$$

In our case we flatten the matrix

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \mapsto \begin{pmatrix} a & b & c & d \end{pmatrix}$$

to improve data locality and cache misses. In this model positions become $(i,j) \mapsto (i + j \cdot L)$ and equivalently $n \mapsto (n \bmod L, \lfloor n/L \rfloor)$.

6. *Metropolis–Hastings*

For our simulations we will utilize the Metropolis–Hastings algorithm.

Algorithm 1 Metropolis–Hastings

```
 1: procedure METROPOLIS–HASTINGS($\mathbf{X}, \mathbf{T}, N$)
 2:     for $i = 0$ to $i = N-1$ do
 3:         Sample $\mathbf{X}'$ according to $\mathbf{T}(\mathbf{X}_i \to \mathbf{X}')$
 4:         if $\mathbf{T}$ is non-symmetric then
 5:             $\mathbf{A}(\mathbf{X}_i \to \mathbf{X}') = \min\left(1, \frac{p(\mathbf{X}')\mathbf{T}(\mathbf{X}' \to \mathbf{X}_i)}{p(\mathbf{X}_i)\mathbf{T}(\mathbf{X}_i \to \mathbf{X}')}\right)$
 6:         else
 7:             $\mathbf{A}(\mathbf{X}_i \to \mathbf{X}') = \min\left(1, \frac{p(\mathbf{X}')}{p(\mathbf{X}_i)}\right)$
 8:         Sample $r$ for $U(0,1)$
 9:         if $r \leq \mathbf{A}(\mathbf{X}_i \to \mathbf{X}')$ then
10:             $\mathbf{X}_{i+1} \leftarrow \mathbf{X}'$
11:         else
12:             $\mathbf{X}_{i+1} \leftarrow \mathbf{X}_i$
```

In our case $\mathbf{T}$ is always symmetric so it reduces to pure Metropolis method. Further, our acceptance $\mathbf{A}$ is dependent on ΔE. We get that

$$r \leq e^{-\beta \Delta E}$$

is considered accepted. As $r \in (0,1)$ we see that $\Delta E \leq 0$ yields $e^{-\beta\Delta E} \geq 1$ meaning that it is always accepted. Further, as ΔE only can attain 5 values, as described in Section II A 2, we can precompute the Boltzmann factors to avoid heavy computations. So we make an array `boltz.factors` that stores these values so they can be reused. Further, this also makes the program be able to compute the total energy without having to go through the entire system each time. It can instead change the energy by ΔE if the spin change is accepted and the same for the magnetization M. So our specialized algorithm based on Algorithm 1 is formulated as follows:

Algorithm 2 Metropolis

```
 1: for step = 0 to $step = N-1$ do                          ▷ $N = L^2$
 2:     $p \leftarrow U\{0,\dots,N-1\}$              ▷ Choose random grid point.
 3:     $s \leftarrow \text{lat}(p)$                              ▷ Spin at position
 4:     i, j $\leftarrow p$                                   ▷ Position in matrix.
 5:     $n \leftarrow \text{lat}(i+1,j)+\text{lat}(i-1,j)+\text{lat}(i,j+1)+\text{lat}(i,j-1)$
 6:     $\text{Factor}_i \leftarrow (s \cdot n + 4)/2$
 7:     $\Delta E \leftarrow 2J\, s\, n$
 8:     $\Delta M \leftarrow -2s$
 9:     accept $\leftarrow (\Delta E \leq 0)$                          ▷ Boolean
10:
11:     if not accept then
12:         $r \leftarrow U(0,1)$                      ▷ Uniform distribution
13:         accept $\leftarrow r \leq \text{boltz.factor}[\text{Factor}_i]$
14:
15:     if accept then
16:         $s \leftarrow -\text{lattice}(i,j)$                       ▷ Flips spin
17:         $E \leftarrow E + \Delta E$
18:         $M \leftarrow M + \Delta M$
```

7. *Parallelization & Multiple walkers*

To reduce the runtime of our simulations, we parallelize the simulation over multiple CPU-threads. Instead of running all cycles in the same simulation we construct multiple independent walkers.

For a given lattice size L and temperature T, we initialize W walkers. We then initialize the lattices randomly for each walker that runs it's own Metropolis algorithm 2. This is to further reduce skewed results from initial conditions. We use burn-in cycles for each walker to insure this.

Algorithm 3 MCMC-Run

```
 1: for $i = 0$ to $i = W - 1$ do                          ▷ Initialize walkers
 2:     Draw seed $s_i$ from master RNG
 3:     Initialize rng$_i$ with $s_i$
 4:     Initialize lattice lat$_i$ from input and spin configuration
 5:     walker$_i \leftarrow$ (rng$_i$, lat$_i$, model)
 6: Start parallelization
 7: for $i = 0$ to $i = W - 1$ do                          ▷ One walker per thread
 8:     (rng, lat, model) $\leftarrow$ walker$_i$
 9:     $E \leftarrow$ total_energy(lat, model)
10:     $M \leftarrow$ total_magnetization(lat)
11:     for $b = 0$ to $b =$ burn_in_cycles $- 1$ do        ▷ Burn in
12:         Metropolis(model, lat, params, rng, $E$, $M$)     ▷ Algorithm 2
13:     $E \leftarrow$ total_energy(lat, model)
14:     $M \leftarrow$ total_magnetization(lat)
15:     for $s = 0$ to $s =$ total_cycles $- 1$ do
16:         Metropolis(model, lat, params, rng, $E$, $M$)
17:         Measurements on the system
```

Each walker collects its own data for both ϵ_i and m_i. This is concatenated after all walkers are finished to calculate the expectation values.

$$\langle X \rangle = \frac{1}{n_{\text{tot}}} \sum_{w=1}^{W} \sum_{i=1}^{n_w} X_i^{(w)}, \quad n_{\text{tot}} = \sum_{w=1}^{W} n_w,$$

where n_W is the number of Metropolis–Hastings cycles per walker and X is an observable of the system. It is worth noting that concatenating the series will give a discontinuity in the ϵ- and m-series. However, this does not affect the expectation values.

We will also benchmark the performance and see how much parallelization speeds up the algorithm. We will do this by measuring the time it takes for n_{threads} threads t_n against the time for one thread t_1.

$$\text{speedup} = \frac{t_1}{t_n}.$$

Ideally, this will yield

$$\text{speedup} = \frac{t_1}{t_1/n} = n_{\text{threads}}.$$

To see the speedup of the algorithm we will only measure the time over the `MCMC_RUN`-Algorithm 3, not the entire program.

8. Pseudorandomization

To draw random variables we use a base seed `seed` = 67. From this base seed we construct a pseudorandom number generator (PRNG) for each walker. For the walker w_i we form a `std::seed_seq` from the pair (`base_seed`, i). We use this to generate `seed`$_i$ that is stored in the seed container. Each walker then initializes its own `std::mt19937` generator as

```
std::mt19937 rng_i(seed_i).
```

This generator is used both to generate a random initial lattice and also for drawing from probability distributions.

C. Other Tools

For plotting we used the Python library `matplotlib` [9], and `numpy` [10] for general calculations.

Statistical analysis and curve fitting were performed with `scipy.stats.linregress` for linear regression and `numpy.polyfit` for the second-order polynomial fit.

For parallelizing the Monte Carlo simulations we used OpenMP [11], by distributing independent walkers over CPU threads.

For reading, writing, and parsing configuration and results files we used the `nlohmann::json` C++ library. [12]

We also used OpenAI's ChatGPT as declared in Appendix C. This was primarily to reduce tedious tasks and polish language and structure.

III. RESULTS AND DISCUSSION

A. Verification for 2×2

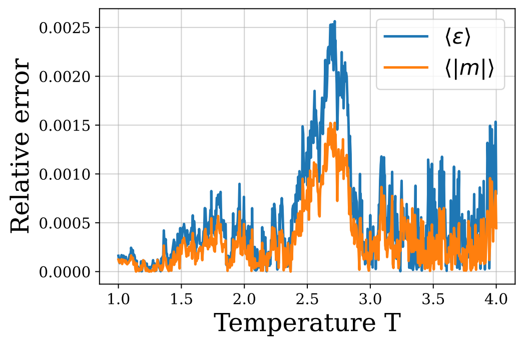


FIG. 2: Relative error for $\langle \epsilon \rangle, \langle |m| \rangle$ plotted against temperature T, each point temperature has 10^6 cycles and started in random initial state.

To verify that our algorithm works we do tests on a 2×2 grid and compare against the analytical results. In

Figure 2 the numerical data is close to the analytical expectation values found in Section II A 5. For $T \in [2.5, 3.0]$ the error spikes, indicative of a critical point.

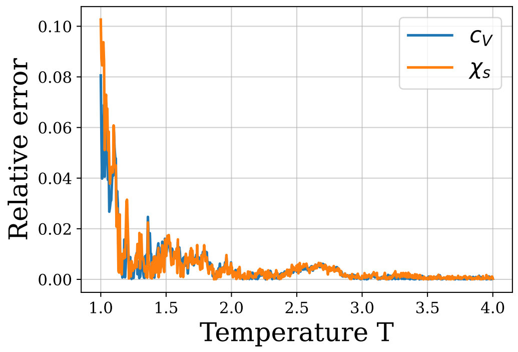


FIG. 3: Relative error for $c_V, \chi_{\rm s}$ plotted against temperature T, each point temperature has 10^6 cycles and started in random initial state.

In Figure 3 there are clear deviations from the analytical answer for low temperatures. If we were interested in simulating lower temperatures, we would need to reconsider our model. It is also worth noting that the actual values of c_V and $\chi_{\rm s}$ are very close to 0, inflating the relative error. There is also again an error spike in $T \in [2.5, 3.0]$, but it is less prominent than the error at low temperatures.

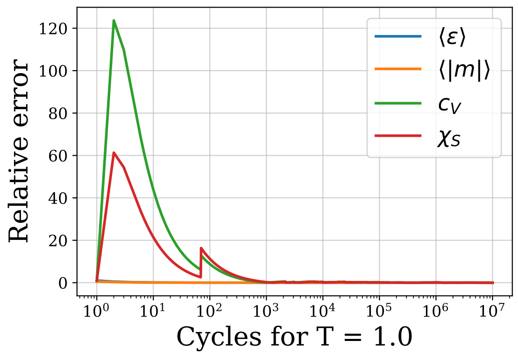


FIG. 4: Relative error for $\langle\epsilon\rangle, \langle|m|\rangle, c_V$ and $\chi_{\rm s}$ plotted against amount of cycles n. For $T = 1.0\, J/k_B$ and random initial state.

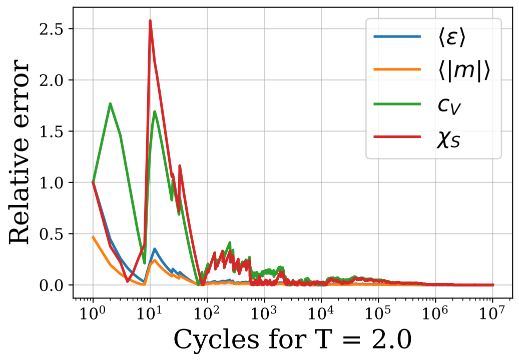


FIG. 5: Relative error for $\langle\epsilon\rangle, \langle|m|\rangle, c_V$ and $\chi_{\rm s}$ plotted against amount of cycles n. For $T = 2.0$ and random initial state.

When looking at how the relative error evolves over multiple cycles for $T = 1.0$ in Figure 4 it seems to be a good approximation around 10^3, but this contradicts Figure 3 as there is a relative error of 0.1 after 10^6 cycles which means for low temperatures cycles will need to be around 10^6 to get a rough estimate, and even higher to get close to expected values. For $T = 2.0$ in Figure 5 the numerical approximation seems to be a good estimation around $10^3 - 10^4$ cycles. This also matches the temperature ranges as the relative error around $T = 2.0$ is around 0.01 for 10^6 cycles.

TABLE III: Simulation vs. analytical with $3 \cdot 10^7$ total cycles and random initial spin configuration and temperature $T = 1.0\, J/k_B$. The relative error is in percentage (%).

Quantity	Simulation	Analytical	Abs. error	Rel. error (%)
$\langle\lvert m\rvert\rangle$	0.998652	0.998661	9.12×10^{-6}	0.00091
$\langle\epsilon\rangle$	-1.99596	-1.99598	2.34×10^{-5}	0.00117
C_V/N	0.032265	0.0320823	1.82×10^{-4}	0.56877
χ/N	0.00404176	0.00401074	3.10×10^{-5}	0.77346

TABLE IV: Simulation vs. analytical with $3 \cdot 10^7$ total cycles with all spins up in initial condition and temperature $T = 1.0\, J/k_B$. The relative error is in percentage (%).

Quantity	Simulation	Analytical	Abs. error	Rel. error (%)
$\langle\lvert m\rvert\rangle$	0.998656	0.998661	4.27×10^{-6}	0.00043
$\langle\epsilon\rangle$	-1.995976	-1.995982	5.89×10^{-6}	0.00029
C_V/N	0.032126	0.0320823	4.33×10^{-5}	0.13498
χ/N	0.00403018	0.00401074	1.94×10^{-5}	0.48470

We see that after even $3 \cdot 10^7$ cycles we still obtain small, but measurable deviations from the analytical solutions. Comparing the result from random initial conditon in Table III to the one with all spins up in Table IV, we also see a difference. As we know that the expected energy per spin should be $\langle\epsilon\rangle \approx -2J$ and magnetization per spin $\langle|m|\rangle \approx 1$, we start the system in a very probable state. However, the relative difference is still of the same magnitude. This implies that the errors may come from finite sampling and dependence in the Markov chain and not an error in the algorithm or floating point errors.

B. Burn-in time

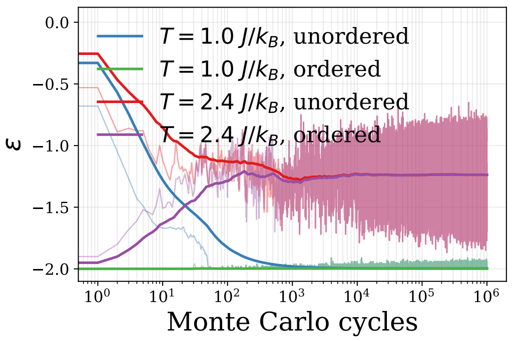


FIG. 6: Monte Carlo cycles against energy per spin ϵ. The thin lines are energy per spin $\langle\epsilon\rangle$. The thick lines are cumulative sums divided by number of cycles. The x-axis is logarithmic. One cycle is $N = L^2$ attempted spin-flips. Only one walker $W = 1$ was utilized.

In Figure 6 we see that after approximately $N = 1{,}000$ cycles all values seem to have stabilized. So for further simulations to ensure stable states we will discard the first 10,000 cycles as burn-in cycles to be on the safe side.

C. Parallelization speed-up

After parallelization we benchmark the runtime of our code.

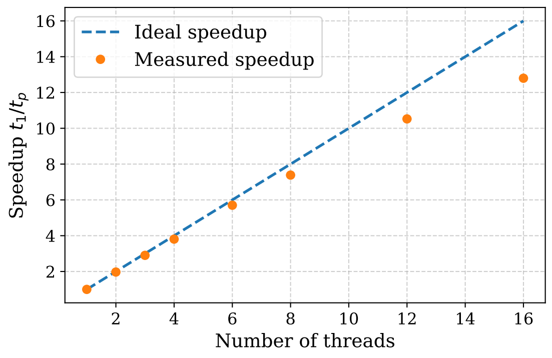


FIG. 7: Speedup t_1/t_n of the `MCMC-RUN`-algorithm. No burn-in cycles and $W = 48$ walkers with 10,000 cycles each for $L = 20$. The dotted line represents ideal speedup $t_1/t_n = n$. The x-axis is the number of threads used. (Ran on AMD Ryzen 7 9800X3D)

As we have parallelized over different walkers it is unnecessary to have more threads available than walkers W. We therefore only have points at $W \mod n_{\text{threads}} = 0$.

For the first additional threads the speedup seems to follow the expected speedup $t_1/t_p = n_{\text{threads}}$ in Figure 7. However, at around $n_{\text{threads}} = 6$ it seems to fall off. In most of the simulations we used 12 treads, which meant we saved time by a factor of 10.5.

Note that the number of threads are not necessarily the ones used, but rather how many that were allocated. On the computer used 16 threads is the maximum. Thus, the operating system and other applications share resources those threads.

D. Probability distributions for $L = 20$

The probability distributions were found with $W = 12$ walkers with 10,000 burn-in cycles and 10^6 cycles per walker.

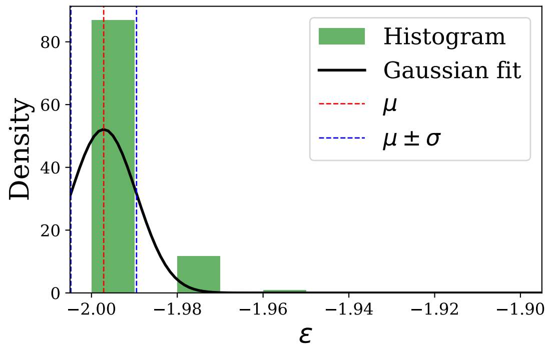


FIG. 8: Density histogram of ϵ-values at $T = 1.0\,J/k_B$ with bin width $4J/L^2 = 0.01J$. The black line is a Gaussian fit around mean value $\mu = -1.9972\,J$ and standard deviation $\sigma = 0.0076\,J$.

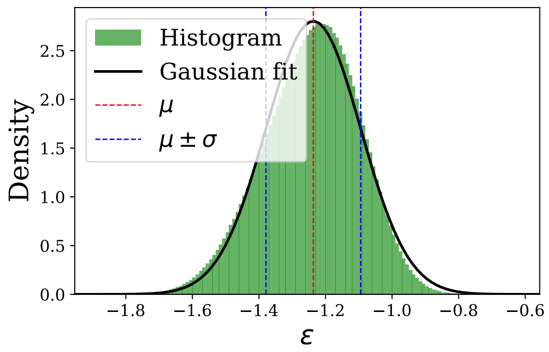


FIG. 9: Density histogram of ϵ-values at $T = 2.40\,J/k_B$ with bin width $4J/L^2 = 0.01J$. The black line is a Gaussian fit around mean value $\mu = -1.2370\,J$ and standard deviation $\sigma = 0.1426\,J$.

In Figure 9 the Gaussian probability distribution seems to be a good fit. There are some noticeable deviations, but this is expected for a finite number of cycles.

For $T = 1.0\,J/k_B$ the histogram fits the Gaussian in Figure 8 poorly. Firstly, because ϵ can not attain values $\epsilon < -2J$ and $\langle\epsilon\rangle \approx -2.0J$. For $T = 1.0\,J/k_B$ we also have that the macrostate $\epsilon = -2J$ is much more likely than any other. This is also seen by the low standard deviation $\sigma = 0.0076\,J$, which is lower than the bin width of $0.01\,J$. For $T = 2.4\,J/k_B$ they are spread more evenly.

The reason for plotting a Gaussian fit is by the central limit theorem. Our ϵ-values are an average of many bonds which are randomly distributed. When these also are weakly correlated, the central limit theorem tells us that the distribution should be approximately normal around $\langle\epsilon\rangle$. This also explains the poor fit for $T = 1.0\,J/k_B$, as the spin configurations are highly correlated.

E. Expectation values for different temperatures

For fixed L, we ran multiple MCMC-cycles over temperatures in the range $T \in [1.0, 5.0]$ to find the critical temperature T_c. Around the range $T \in [2.1, 2.4]$ we made sure to do $n = 10^6$ MCMC-cycles for each point to get the best estimate for $\hat{T}_c$. When we recognized a peak as in Figure 15, we narrowed down the T -range around the peak. Then repeated the procedure until the polynomial fit and individual uncertainty of $\hat{T}_c^{(c_V)}$ and $\hat{T}_c^{(\chi_s)}$ was satisfactory. For lager lattice sizes we did not narrow the T -range as far. But overall, every repetition brought us closer to Onsager's value for $T_c(L = \infty)$.

However, simulating outside the range $T \in [2.1, 2.4]$ was purely for visualization purposes so mostly the points are found with $n = 10^5$ cycles to reduce runtime. We also ran $W = 12$ walkers for all simulations at all intervals.

1. 20×20 *lattice*

As the system's properties behave similarly for all L, we choose $L = 20$ as an example. For these simulations the procedure was equal to all other lattice sizes.

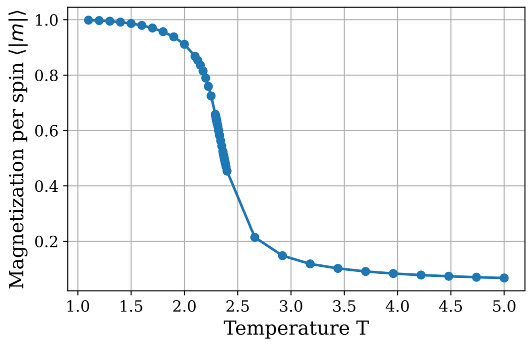


FIG. 10: Reduced average absolute magnetization $\langle|m|\rangle$ against temperature T for $L = 20$.

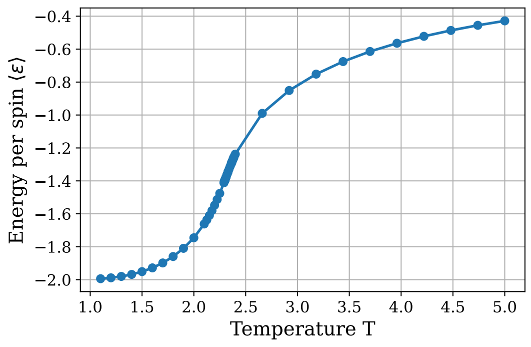


FIG. 11: Reduced average energy $\langle\epsilon\rangle$ against temperature T for $L = 20$.

Both the energy $\langle\epsilon\rangle$ and magnetization $\langle|m|\rangle$ per spin exhibits a sigmoidal curve in Figures 10 and 11. The areas with more dense data points is where we searched for the critical temperature c_V and susceptibility χ_s.

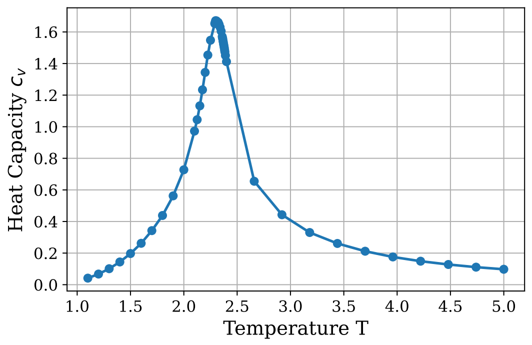


FIG. 12: Specific heat capacity c_V against temperature T for lattice length $L = 20$.

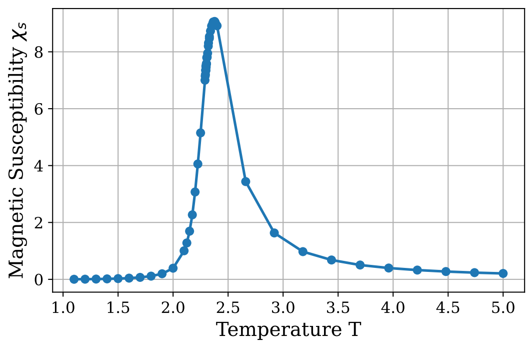


FIG. 13: Specific susceptibility χ_s against temperature T for lattice length $L = 20$.

Both the heat capacity c_V and susceptibility χ_s per spin attain their lower values at lower and higher temperatures. However, centered at around $T \approx 2.4$ there is a spike for Figure 12 and 13. This is around the critical temperature and is where we do the polynomial fits.

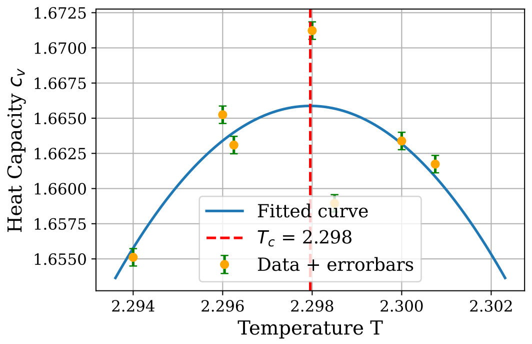


FIG. 14: 2nd order polynomial fit around maximum c_V from Figure 12.

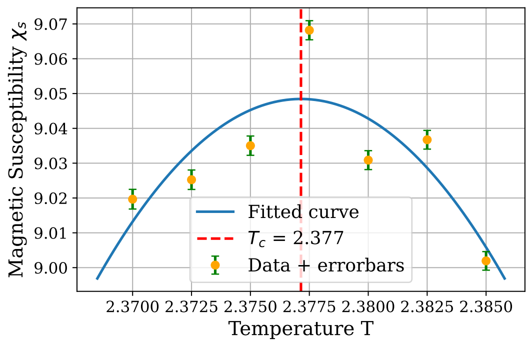


FIG. 15: 2nd order polynomial fit around maximum χ_s from Figure 13.

We see from Figures 14 and 15 that the data points seem to deviate randomly from the fitted curve. However, most points plus uncertainties do not overlap with the fit. This either means that it is a bad fit or that our uncertainty is underestimated, since we ignored autocorrelation between samples. The share amount of cycles ($M = 10^6$ cycles $\times$ $W = 12$ walkers) gives an expectation of a small uncertainty. Considering that we clearly see a rounded peak in both c_V and χ_s from Figures 12 and 13 a 2nd order polynomial fit should however, be a good approximation for data points close to the peak.

This indicates that our uncertainties are underestimated from our assumption of independent states. As mentioned in II B 3, we should in fact consider the effective independent states $n_{\text{eff}} < n$. This would give a larger and more correct approximation for the uncertainty of the individual data points.

2. *Combined plots*

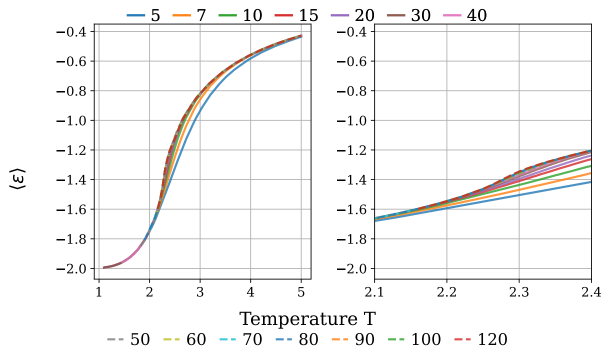


FIG. 16: Average energy per spin $\langle\epsilon\rangle$.

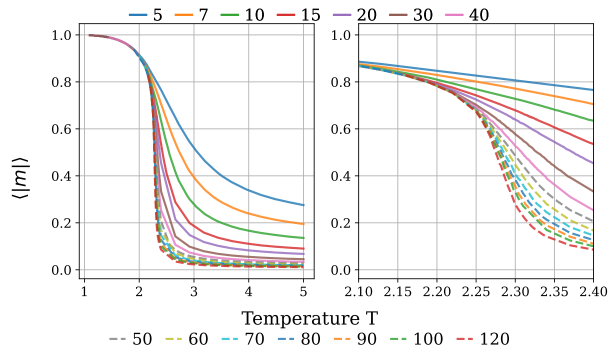


FIG. 17: $\langle|m|\rangle$ against temperatures T.

Both, Figure 16 and 17 display a sigmoidal curve, but in the opposite direction. For lower temperatures the energy is minimized, while the magnetization is maximized. This makes sense as the spins tend to be in the same direction to minimize the energy. However, reaching the critical temperature this turns and the energy gets higher and the magnetization goes towards 0. The slopes also increases with L and we see a more rapid change in the systems properties. In Figure 17 the drop is expected from a ferromagnet at the Curie temperature T_C as the it turns into a paramagnet.

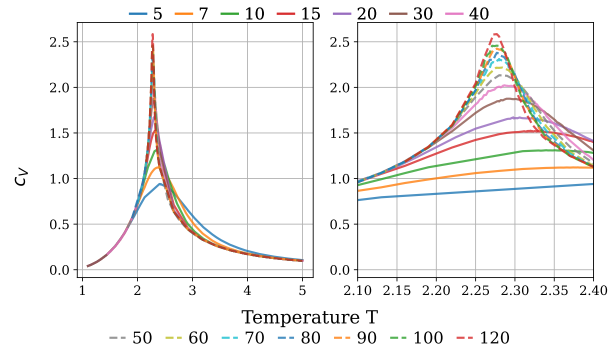


FIG. 18: Specific heat capacity c_V against temperature T. The different lines represent different lattice sizes.

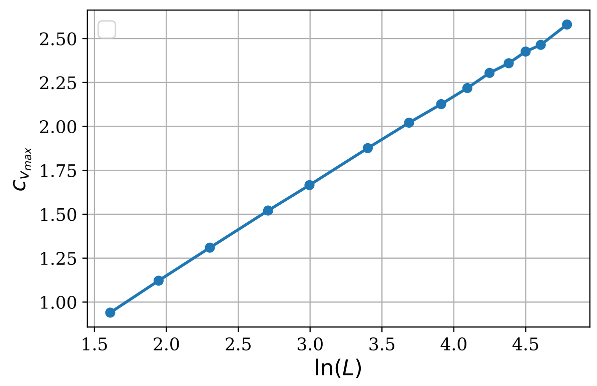


FIG. 19: Estimated specific heat capacity at the critical point $\hat{c}_{V,c}$ plotted against $\ln L$.

Figure 18 shows specific heat capacity against temperature, for big L there is a clear indication of divergence which is is expected during a phase transition, the relation between the maxima of c_V and L is shown in Figure 19. The figure shows a proportionality between c_V and $\ln L$ which is exactly what Equation (4) predicted.

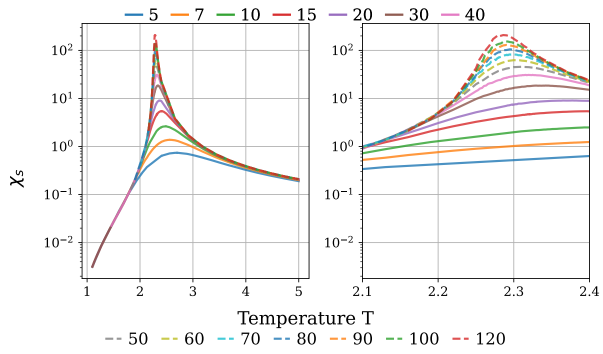


FIG. 20: Specific susceptibility χ_s against temperature T. The different lines represent different lattice sizes. Logarithmic y-axis.

From Figure 20 the divergence at T_c as $L \to \infty$ is clear and expected when magnetization drops, the maxima follows the relation given in Equation (3). This is clear by the fact that the y-axis is logarithmic. The tail to the left is exactly what Curie-Weiss law and Equation (2) predicts as they are meant to explain the behavior for temperature at the critical point and higher.

One also sees that for higher L-values, the T-series are truncated. This is due to much noise for lower amount of cycles. As it also takes much longer time to run, we have ignored these points.

F. Comparison with Onsager's result

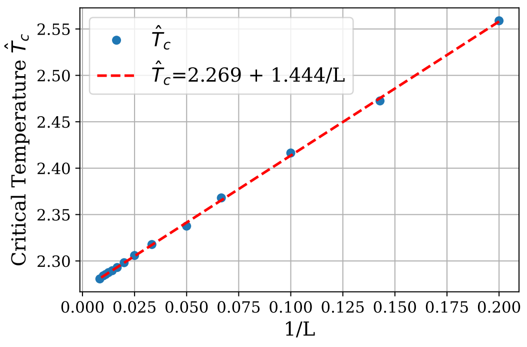


FIG. 21: Linear fit of $1/L$ against the found critical temperature $\hat{T}_c(L)$. We have slope $m = 1.444 \pm 0.009\, J/k_B$ and intercept $b = 2.269 \pm 0.001\, J/k_B$. Individual $T_c^{(c_V)}$ and $T_c^{(\chi_s)}$ and measured maximum values of c_V and χ_s is to be found in table V in appendix B.

In Figure 21 we have done a linear fit on $1/L$ against the critical temperature $\hat{T}_c(L)$ using average of c_V and χ_s as in equation (5). We then get the limit

$$\lim_{L\to\infty} \hat{T}_c(L) = 2.26917 \pm 0.00014 J/k_B,$$

compared to Onsager's analytical result of $T_c \approx 2.269185311421\, J/k_B$. The result should probably be rounded off by one decimal, because of the uncertainty, but we wanted to show how close we came. Either way, the relative error became

$$\frac{\hat{T}_c - T_c}{T_c} \approx 6.75 \cdot 10^{-6}.$$

The figure also shows us that the critical temperature decrease slightly with increasing L. This is consistence with earlier observations, with shifting peaks in c_V and χ_s. For smaller lattice sizes of c_V and χ_s, the peaks were more spread out. For larger sizes they became more packed together, because peaks grew sharper and higher. As we approach $L = \infty$ we expect both the peaks to lie on top of each other on Onsager's result. This trend is visible in both Figure 20 and 18. The spread in the peaks for $L = 5$ is ≈ 0.28, and for $L = 120$ is ≈ 0.01.

Either way we came quite close to Onsager's value. By narrowing the temperature range and increasing the range of lattice sizes we came closer and closer. To reach the 12th decimal, the trend seems to not need the largest lattice sizes, but rather sharper values in the range we are already looking at.

IV. CONCLUSION

In this work, we developed a Metropolis MCMC algorithm for analyzing the 2D ferromagnetic Ising model. We ran simulations on a square lattice and studied system properties like energy, magnetization, heat capacity and magnetic susceptibility. This was used to analyze phase transitions and behavior across different temperatures and different lattice sizes.

First, we validated the code against analytical expressions for a 2×2 system. The results agreed with the theory, but with some caveats. For instance, we noticed larger deviations for low temperatures $T \approx 1 J/k_B$. We also saw the same around the critical temperature. However, the relative errors all below 1% for all values after 10^6 cycles.

Having confirmed that our method gave good estimations for the 2×2 model, we studied burn-in time for $L = 20$. We were conservative and chose 10^4 cycles as all values seemed to be stabilized. This was used for all further simulations. When increasing the lattice size, we saw that our code needed to run quicker. Therefore, we implemented parallelization using OpenMP and benchmarked the runtime. For further simulations we decided to use 12 threads giving an ≈ 10.5-times speedup. This also gave us more accurate results as we initialized independent walkers.

Computing the energy for $L = 20$ we saw a clear temperature dependence. For $T = 1.0\, J/k_B$, we got a histogram with $\langle \epsilon \rangle = -1.9972\, J$ with a low standard deviation $\sigma \approx 0.0076\, J$. For $T = 2.4\, J$, we got a clearly Gaussian probability distribution for ϵ centered at $\langle \epsilon \rangle = -1.2370\, J$ with standard deviation $\sigma = 0.1426$.

Looking at multiple lattices with different L and temperatures T we saw very similar behavior. Both the energy and magnetization followed sigmoidal curves over T for fixed L. They were however opposite: lower energies/temperatures gave high magnetization while higher energies/temperatures gave lower magnetization. This was predicted by the theory where we hypothesized that higher external temperatures turns the system into a paramagnet. The rate was also more extreme for higher L as predicted. For higher L, the slope of $\langle |m| \rangle$ was much higher and the value of χ_s and c_V seemed to diverge as $L \to \infty$. The critical temperatures were also shifted leftwards for higher L. We used this to compare to Lars Onsager's result.

Doing a linear fit of the critical temperature against L gave us an estimated $\hat{T}_c(L = \infty) = 2.26917 \pm 0.00014\, J/k_B$. This was a relative error of $\leq 10^{-5}$ compared to Onsager's result.

For further improvements of our results, we should revisit our approach to uncertainties. The uncertainty of the observables were vastly underestimated due to an overestimation of independent cycles. As discussed this can be fixed by finding an autocorrelation function and reduce the number of efficient cycles to n_{eff}. It could also be worth looking into if it would then be more efficient to introduce more walkers as they are more independent. This would however require more burn-in cycles so we would have to compromise between amount of cycles and amount of walkers.

We could also generalize the Ising model by adding the opportunity to have other shapes than squares or more dimensions. Another important addition could be an external magnetic field. This would however require us to vary both temperature and magnetic field to find critical points adding more complexity. As we were only interested in qualitative behavior and comparison to Lars Onsager's result without an external field, this was not necessary.

Our work shows that Markov Chain Monte Carlo methods do a good job of capturing the behavior of a two-dimensional ferromagnetic Ising model. We get results that match the theory with high precision. The qualitative behavior also matches what we expect from known thermodynamic identities. Expanding on the framework can be used to find values for actual systems and not just the Ising model. However, for a ferromagnet, our results are a good indicator of the physical behaviour.

[1] Wikipedia contributors. Ising model. https://en.wikipedia.org/wiki/Ising_model, 2025. Accessed: 2025-11-17.

[2] Ta2o. 2d ising model on lattice (svg). https://commons.wikimedia.org/wiki/File:2D_ising_model_on_lattice.svg, 2025. Wikimedia Commons. CC BY 4.0. Accessed: 2025-10-25.

[3] Morten Hjorth-Jensen. Computational physics. Lecture notes, Department of Physics, University of Oslo, 2015. Fall 2015 edition.

[4] Anders Kvellestad. Fys3150 project 4. https://anderkve.github.io/FYS3150/book/projects/project4.html, 2025. Accessed: 2025-10-25.

[5] Wikipedia contributors. Curie–weiss law. https://en.wikipedia.org/wiki/Curie%E2%80%93Weiss_law, 2025. Accessed: 2025-11-18.

[6] Wikipedia contributors. Square lattice ising model. https://en.wikipedia.org/wiki/Square_lattice_Ising_model, 2025. Accessed: 2025-11-12.

[7] Anders Kvellestad and collaborators. Fys3150 lecture notes. https://github.com/anderkve/FYS3150/tree/master/lecture_notes. Accessed: 2025-10-25.

[8] Andy Jones. Effective sample size. https://andrewcharlesjones.github.io/journal/21-effective-sample-size.html, 2021. Accessed: 2025-11-16.

[9] John D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[10] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[11] OpenMP Architecture Review Board. *OpenMP Application Programming Interface*, version 5.2 edition, 2021. Accessed: 2025-10-23.

[12] Niels Lohmann. Json for modern c++. https://github.com/nlohmann/json. MIT License. Accessed: 2025-10-23.

Appendix A: Derivations

In this appendix we derive analytical expressions for the Ising model and their uncertainties.

1. Analytical values 2×2

With energies E_i and degeneracies g_i, $E = -8J$ (deg. 2), $E = 0$ (deg. 12), $E = +8J$ (deg. 2), the partition function is

$$Z = \sum_{\mathbf{s}} e^{-\beta E(\mathbf{s})} = \sum_{i}^{N} g_i e^{-\beta E_i} = 2\, e^{+8\beta J} + 12 + 2\, e^{-8\beta J} = 12 + 4\cosh(8\beta J).$$

For the expectation value of the energy we use the same degeneracies as for the partition function. Which gives

$$\langle E \rangle = \sum_{\mathbf{s}} E(\mathbf{s}) p(\mathbf{s};T) = \sum_{\mathbf{s}} E(\mathbf{s}) \frac{e^{-\beta E(\mathbf{s})}}{Z} = \frac{1}{Z}\left(1\cdot(-8J)e^{8\beta J} + 4\cdot 0 + 2\cdot(8J)e^{-8\beta J} + 4\cdot 0 + 4\cdot 0 + 1\cdot(-8J)e^{8\beta J}\right)$$
$$= \frac{16J}{Z}\left(e^{-8\beta J} - e^{8\beta J}\right) = -\frac{32J}{Z}\sinh(8\beta J),$$

and

$$\langle E^2 \rangle = \sum_{\mathbf{s}} (E(\mathbf{s}))^2 \frac{e^{-\beta E(\mathbf{s})}}{Z} = \frac{1}{Z}\left(1\cdot(-8J)^2 e^{8\beta J} + 4\cdot 0 + 2\cdot(8J)^2 e^{-8\beta J} + 4\cdot 0 + 4\cdot 0 + 1\cdot(-8J)^2 e^{8\beta J}\right)$$
$$= \frac{128J^2}{Z}\left(e^{8\beta J} + e^{-8\beta J}\right) = \frac{256J^2}{Z}\cosh(8\beta J).$$

Dividing by N when gives

$$\langle \epsilon \rangle = \frac{\langle E \rangle}{N} = -\frac{32J}{NZ}\sinh(8\beta J),$$
$$\langle \epsilon^2 \rangle = \frac{\langle E^2 \rangle}{N^2} = \frac{256J^2}{N^2 Z}\cosh(8\beta J).$$

Magnetization have energies and degeneracies is from Table II, which gives the expectation values

$$\langle |M| \rangle = \sum_{\mathbf{s}} |s_i| p(\mathbf{s};T) = \frac{1}{Z}\sum_{\mathbf{s}} |s_i| e^{-\beta E(\mathbf{s})} = \frac{1}{Z}\left(4e^{+8\beta J} + 4\cdot 2 + 4\cdot|-2| + |-4|e^{+8\beta J}\right) = \frac{8}{Z}(e^{+8\beta J} + 2),$$

and

$$\langle M^2 \rangle = \frac{1}{Z}\sum_{\mathbf{s}} s_i^2 e^{-\beta E(\mathbf{s})} = \frac{1}{Z}\left(4^2 e^{+8\beta J} + 4\cdot 2^2 + 4\cdot(-2)^2 + (-4)^2 e^{+8\beta J}\right) = \frac{32}{Z}(e^{+8\beta J} + 1).$$

Writing it per spin gives

$$\langle |m| \rangle = \frac{8}{NZ}(e^{+8\beta J} + 2),$$
$$\langle m^2 \rangle = \frac{32}{N^2 Z}(e^{+8\beta J} + 1).$$

2. Heat capacity and susceptibility

The heat capacity and susceptibility is calculated using the expectation values of the energy and magnetization respectively.

$$c_V = \frac{C_V}{N} = \frac{1}{N}\frac{1}{k_B T^2}\Big(\langle E^2 \rangle - \langle E \rangle^2\Big) = \frac{N}{k_B T^2}\Big(\langle \epsilon^2 \rangle - \langle \epsilon \rangle^2\Big)$$

and

$$\chi_{\mathrm{s}} = \frac{\chi}{N} = \frac{1}{N}\frac{1}{k_B T}\Big(\langle M^2 \rangle - \langle |M| \rangle^2\Big) = \frac{N}{k_B T}\Big(\langle m^2 \rangle - \langle |m| \rangle^2\Big).$$

In addition, the expressions are valid for all lattice sizes.

3. Uncertainties

We have that the uncertainty for c_V, χ_s is given by $\mathrm{Var}(F) \approx \nabla f^\top \mathrm{Cov}(\mu) \nabla f$. We can also use that $\mathrm{Cov}(\mu_1, \mu_1) = \mathrm{Var}(\mu_1)$, $\mathrm{Cov}(\mu_2, \mu_2) = \mathrm{Var}(\mu_2)$ and $\mathrm{Cov}(\mu_1, \mu_2)$. For c_V this gives

$$\nabla c_V = \frac{N}{k_B T^2}(1, -2\langle \epsilon \rangle).$$

Then we get

$$\mathrm{Var}(c_V) \approx \left(\frac{N}{k_B T^2}\right)^2 \Big[\mathrm{Var}(\langle \epsilon^2 \rangle) + 4\langle \epsilon \rangle^2 \mathrm{Var}(\langle \epsilon \rangle) - 4\langle \epsilon \rangle \mathrm{Cov}(\langle \epsilon^2 \rangle, \langle \epsilon \rangle)\Big].$$

We further have that

$$\mathrm{Cov}(\langle \epsilon^2 \rangle, \langle \epsilon \rangle) = \frac{1}{n}\left(\langle \epsilon^3 \rangle - \langle \epsilon^2 \rangle \langle \epsilon \rangle\right)$$

and

$$\mathrm{Var}(\langle X \rangle) \approx \frac{1}{n}\mathrm{Var(X)}$$

We further remember that $e_n = \mathrm{Var}(\epsilon^n)$ and $v_n = \mathrm{Var}(|m|^n)$, giving us the uncertainty:

$$\delta(c_V) = \Big(\frac{N}{k_B T^2}\Big)\sqrt{\frac{e_2}{n} + 4\langle \epsilon \rangle^2 \frac{e_1}{n} - 4\langle \epsilon \rangle \frac{\mathrm{Cov}(\epsilon^2, \epsilon)}{n}}.$$

Repeating the process for χ_s, we get:

$$\mathrm{Var}(\chi_s) \approx \left(\frac{N}{k_B T}\right)^2 \Big[\mathrm{Var}(\langle m^2 \rangle) + 4\langle |m| \rangle^2 \mathrm{Var}(\langle |m| \rangle) - 4\langle |m| \rangle \mathrm{Cov}(\langle m^2 \rangle, \langle |m| \rangle)\Big],$$

giving us the uncertainty:

$$\delta(\chi_s) = \Big(\frac{N}{k_B T}\Big)\sqrt{\frac{v_2}{n} + 4\langle |m| \rangle^2 \frac{v_1}{n} - 4\langle |m| \rangle \frac{\mathrm{Cov}(m^2, |m|)}{n}}.$$

Appendix B: Expectation value table

TABLE V: Estimated critical temperatures and maximum values of heat capacity and susceptibility where the polynomial fit originated around for different lattice sizes L.

L	$\hat{T}_c^{(C_V)}$	$\hat{T}_c^{(\chi_s)}$	$c_V^{\max}$	$\chi_s^{\max}$
5	2.41646817 ± 0.0001	2.7012 ± 0.0001	0.9400 ± 0.0003	0.7392 ± 0.0002
7	2.37959576 ± 0.0007	2.5653 ± 0.0004	1.1219 ± 0.0004	1.3764 ± 0.0004
10	2.34189739 ± 0.0008	2.4908 ± 0.0004	1.3097 ± 0.0005	2.6275 ± 0.0008
15	2.31992884 ± 0.0004	2.4160 ± 0.0002	1.5211 ± 0.0006	5.4269 ± 0.0016
20	2.29795756 ± 0.0007	2.3771 ± 0.0008	1.6659 ± 0.0006	9.0484 ± 0.0027
30	2.29312263 ± 0.0004	2.3423 ± 0.0006	1.8764 ± 0.0007	18.4973 ± 0.0056
40	2.28747492 ± 0.0003	2.3243 ± 0.0007	2.0212 ± 0.0008	30.7596 ± 0.0092
50	2.28462345 ± 0.0007	2.3117 ± 0.0003	2.1268 ± 0.0008	45.5096 ± 0.0140
60	2.28221185 ± 0.0004	2.3037 ± 0.0005	2.2185 ± 0.0009	62.6533 ± 0.0201
70	2.28058891 ± 0.0005	2.2985 ± 0.0006	2.3046 ± 0.0009	82.5050 ± 0.0249
80	2.27940604 ± 0.0019	2.2956 ± 0.0005	2.3596 ± 0.0009	104.4012 ± 0.0316
90	2.27860457 ± 0.0005	2.2922 ± 0.0004	2.4262 ± 0.0009	128.7596 ± 0.0404
100	2.27680091 ± 0.0005	2.2912 ± 0.0007	2.4643 ± 0.0010	152.6723 ± 0.0481
120	2.27543102 ± 0.0003	2.2859 ± 0.0008	2.5801 ± 0.0010	208.1590 ± 0.0657

Appendix C: Declaration of Use of Generative AI

In this scientific work, generative artificial intelligence (AI) has been used. All data and personal information have been processed in accordance with the University of Oslo's regulations, and we, as the authors of the document, take full responsibility for its content, claims, and references. An overview of the use of generative AI is provided below.

Summary

- **Tool(s) used:** OpenAI ChatGPT (GPT-5 & GPT-5.1), https://chatgpt.com/
- **Use:**
  - Generating boilerplate code like plotting with `matplotlib`.
  - Generating Makefiles.
  - Formatting README.md.
  - Checking language for clarity and grammar and general proof reading.
  - Generating brief code documentation, comments, and variable names for better readability in adherence to conventions.
  - Creating tables in proper tex-format.
  - Brainstorming ideas for optimizing code for efficiency.