The Worm Algorithm

physics760: Computational Physics
Final Project

Ajay S. Sakthivasan Dongjin Suh

Universität Bonn

March 15, 2023

Presentation Outline

Presentation Outline

Introduction

- The Metropolis algorithm is a widely used Monte Carlo method for the Ising model.
- However we face the problem of critical slowing down.
- Prokof'ev and Svistunov proposed an alternative update algorithm called the Worm Algorithm (WA).
- WA preserves the local nature of the update step, but achieves a very small dynamical exponent.

Presentation Outline

The Ising model

- A mathematical model to understand the behaviour of systems phase transitions, like ferromagnetic materials
- 2D Ising model: Magnetic system consisting of interacting spins on a two-dimensional lattice

Hamiltonian:

$$H = J \sum_{\langle i,j \rangle} s_i s_j - h \sum_i s_i$$

Partition function:

$$Z = \sum_s \mathrm{e}^{-\beta H} = \sum_s \mathrm{e}^{-[-\beta J \sum_{\langle i,j \rangle} s_i s_j - \beta h \sum_i s_i]}$$

Physical observables

Describe the properties of the system change at phase transition

- Magnetisation per spin

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

N replaced by L^2, where L is the lattice length and σ the spin.

- Energy per site

$$E = \frac{1}{N}H \tag{2}$$

- Susceptibility

$$\chi = (k_B\beta) \cdot (\langle M^2 \rangle - \langle M \rangle^2) \tag{3}$$

- Specific heat

$$C = (k_B\beta)^2 \cdot (\langle E^2 \rangle - \langle E \rangle^2) \tag{4}$$

- Autocorrelation time – Dynamical Exponent (z)

$$\tau \approx L^z \text{ for large J and } \beta \tag{5}$$

Presentation Outline

Metropolis-Hastings Algorithm

- This is a Monte Carlo simulation method.
- Generate samples from a probability distribution.
- Iterative update of the system with the Accept-Reject method.
- Implement Metropolis-Hastings method.
  1. Random configuration of an N $\times$ N lattice.
  2. Flip the spin at the site.
  3. Calculate the energy cost ΔE of the flip.
  4. The reject/accept step.
- Repeat these steps for desired number of times and measure the observables at every iteration.

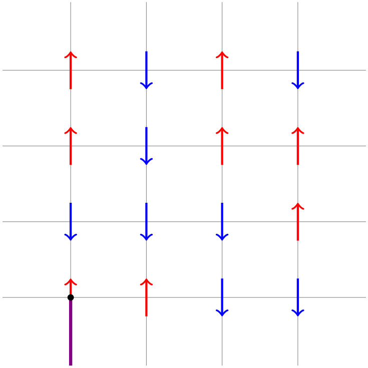
The Worm Algorithm

- The worm algorithm is an alternative to the standard Metropolis algorithm.
- The original implementation by Prokof'ev and Svistunov is roughly as follows:
  1. Start with an arbitrary lattice configuration (with no starting bonds between the sites).
  2. Select an arbitrary point as $i_1 = i_2$.
  3. Grow the worm:
     1. When $i_1 \neq i_2$, choose to move i_1 and create or erase bond line between the sites.
     2. When $i_1 = i_2$, choose with probability 0.5 to start with a new site or to move i_1.
     3. The acceptance probabilities of the moves are given by solutions to the balance equation.
     4. Collect statistics for various quantities and proceed to moving the worm.
  4. Repeat the steps desired number of times.

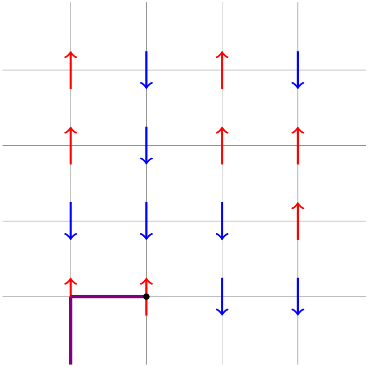
- There are many variations of the worm algorithm.
- Our first implementation of the worm algorithm is as follows:
  1. Start with an arbitrary lattice configuration.[1]
  2. Select an arbitrary point to create the worm.
  3. Grow the worm:
     1. Choose a random direction to move.
     2. If the new point is of the same spin as the old point, add it to the worm with probability 1.
     3. If not, perform a Metropolis-like check. If a flip is favourable, flip and add
     4. This way, we create a worm with equal spins.
     5. Break when the worm head meets its tail or if a flip is no longer favourable.
  4. Measure the observables.
  5. Carry out the desired number of iterations, with a new worm every time.
- There are a few caveats.

[1] We started with an "almost" cold start.

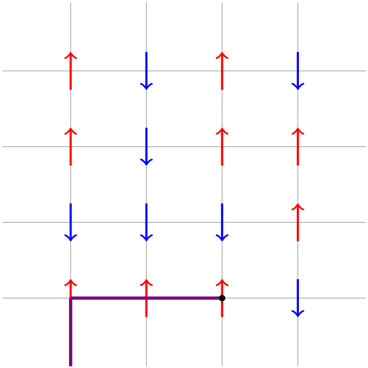
- Given below is a worm at an intermediate step of its growth.
- The lattice below is a small part of the total lattice.



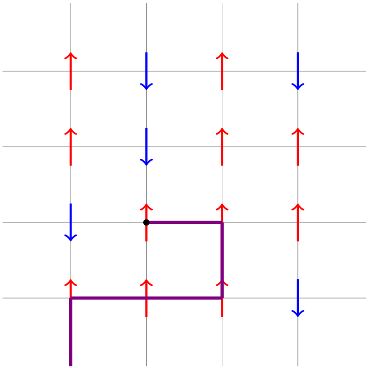
- The worm decides to move right.
- The new site has the same spin as the old site.



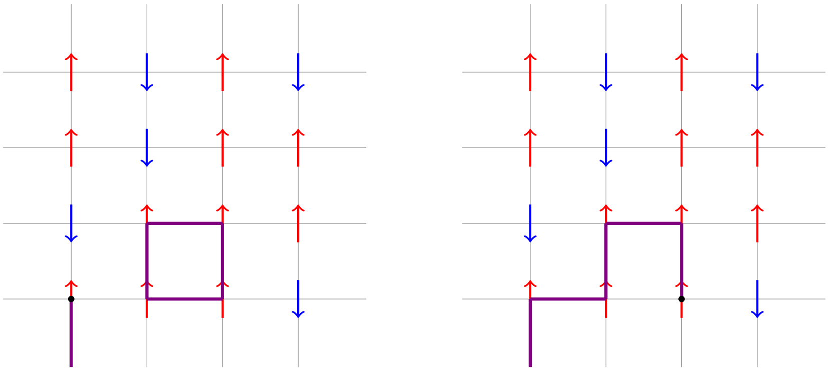
- The worm decides to move right.
- The new site has the opposite spin as the old site.
- Now we perform a metropolis-like check and decide whether to flip or not.



- The worm after a few more steps.



- Caveat: If the worm tries to move to a point that is already a part of the worm, we need to update the head accordingly.
- We want only an even number of bonds between the sites.
- We ensure this by choosing the new head and breaking an old bond appropriately (with equal probability).



- Caveat: If the worm tries to leave the lattice, let it choose a different direction.
- Caveat: The worm cannot be bigger than the total number of lattice sites.
- The worm dies if it hits its tail or if a new flip is no longer favourable.
- Update the observables and start with a new worm.

- **Problem**: This variation of the algorithm did not provide the expected behaviour at low inverse temperatures.
- **Reason**: Say we start with a random site. A flip is always favourable, and a worm keeps growing until breaking conditions are met.
- **Attempted Solution**: Try worms of alternating spins instead of same spins.
- This did solve the problem and resulted in the expected behaviour (including critical point).
- We also tried a mix of the two variations by introducing a cut-off.

How does this differ?

- We always choose to kill the worm when the head and tail meet.
- We implemented a Metropolis-like check for acceptance probabilities.
- Observables are calculated from the lattice configuration at the end of a worm's life.

- In the original implementation, this was done with a probability 0.5.
- Acceptance probabilities are calculated based on the bond configuration.
- Observables are calculated directly from the bond configuration after every time the worm moves.

Error Analysis

- Bootstrap method was used for error analysis.
- The idea is roughly as follows – we resample from the original sample to create multiple samples.
- For each sample, the statistic of interest is then studied.
- The standard deviation of the bootstrap distribution is an estimate of the standard error.
- This method was used to estimate the errors in the quantities in our results.

Presentation Outline

Algorithm behaviour – Metropolis

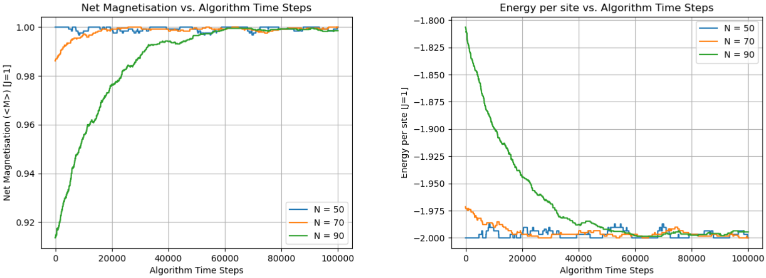


Figure 1: Metropolis behaviour studied with 100000 iterations of the algorithm with 30000 burn-in iterations, $J = 1$ and $\beta = 1$

Algorithm behaviour – Worm Algorithm

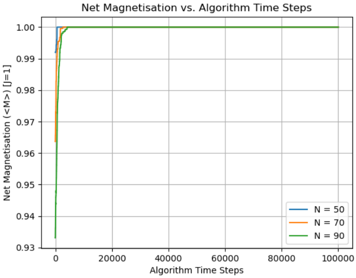


Figure 2: Worm behaviour studied with 100000 iterations of the algorithm with 30000 burn-in iterations, J = 1 and $\beta = 1$

- Both the cases were studied with the same initial configurations – "almost" cold starts closer to spins $= +1$ and -1 respectively.
- We immediately notice that the worm algorithm equilibrates much faster.
- Lattice size doesn't significantly affect equilibration in the case of WA.

Net Magnetisation

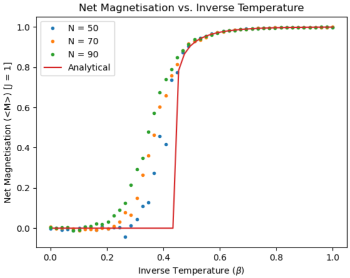


Figure 3: Metropolis algorithm: Behavior of net magnetisation

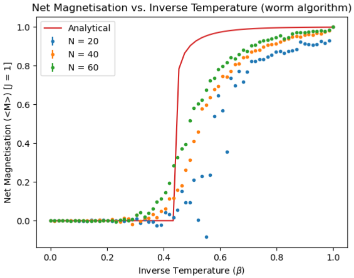


Figure 4: Worm algorithm: Behavior of net magnetisation

- WA was studied for smaller lattices – computationally more expensive when bootstrapping was included.
- We notice worm algorithm is much smoother around the critical points, even for smaller lattice sizes.
- However, it is a bit far from the analytical solution.
- As discussed, we experimented with two different implementations of worm algorithms – this plot corresponds to alternating spins.
- The other variation produced more exact result for high inverse temperature.

Susceptibility and Heat Capacity – Metropolis

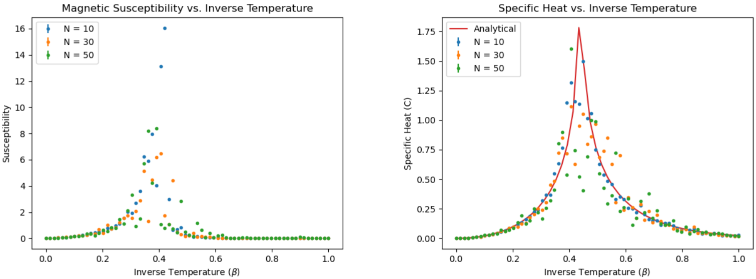


Figure 5: Metropolis Algorithm: Behaviour of Susceptibility and Heat Capacity

Susceptibility and Heat Capacity – Worm Algorithm

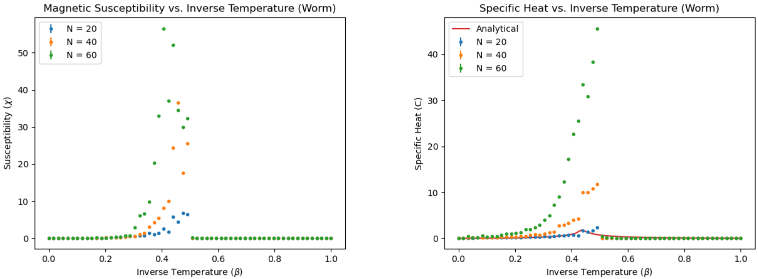


Figure 6: Worm Algorithm: Behaviour of Susceptibility and Heat Capacity

- Again, the worm algorithm produced less sporadic points compared to Metropolis.
- However, we faced an issue with normalisation.

Autocorrelation time – Dynamical Critical Exponent

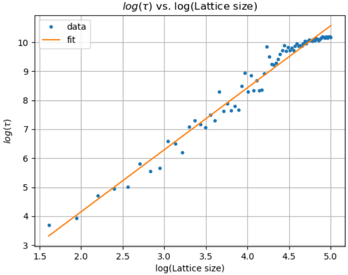


Figure 7: Metropolis Algorithm: Behaviour of Autocorrelation Time

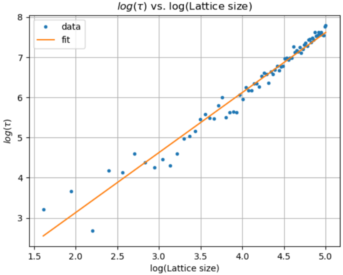


Figure 8: Worm Algorithm: Behaviour of Autocorrelation Time

- Lattice sizes from 5 to 149 were considered for autocorrelation times.
- The autocorrelation times were calculated, and then a $ln - ln$ plot was made.
- The plots here correspond to a dynamical critical exponent of 2.13 and 1.49 for Metropolis and WA, respectively.
- 20 different runs were carried out for autocorrelation time. Dynamical exponent had a range $2.1 - 2.21$ and $1.46 - 1.55$ for Metropolis and WA.
- Statistical analysis was not considered, since 20 runs are not enough.

Presentation Outline

Discussion

- The algorithm is functional to the extent that the results are not unreasonable.
- The dynamical critical exponent was reduced with WA, but still higher than the original work.
- In the original implementation, the dynamical exponent was reported as 0.25.
- We believe this is mainly due to our decision to kill the worm and using a Metropolis-like check.
- However, this hypothesis needs to tested (maybe in the future).
- It should also be noted that lattice sizes up to 512 were considered in the original work.
- Asymptotic result $\implies$ we might get a better result if we simulate larger lattices.

- When it comes to physical observables, the plots showed critical behaviour.
- However, they were not exact compared to analytic results.
- There is also an issue of normalisation, which, for some reason, proved to be very elusive in the case of WA.
- Primary goal for future work, replace with the variation in the original work.
- Measure observables using bond configurations.
- This should further improve our results.
- 3D Ising was not studied due to time constraints.
- Another course of work – Implement the algorithm for 3D Ising model.

Presentation Outline

Summary

- To summarise, we studied the worm algorithm in this project.
- This was then used to simulate the 2D Ising model.
- Various observables and the dynamical critical exponent were calculated.
- The corresponding results were then compared with Metropolis algorithm.
- Shortcomings and problems with the implementation were realised, and possible solutions are suggested.
- Things that were left out due to time constraints could be studied in the near future.

Thanks for your time!

References I

[1] N. Prokof'ev and B. Svistunov, "Worm algorithms for classical statistical models", Physical Review Letters **87**, 10.1103/physrevlett.87.160601 (2001).

[2] N. Prokof'ev, B. Svistunov, and I. Tupitsyn, ""worm" algorithm in quantum monte carlo simulations", Physics Letters A **238**, 253–257 (1998).

[3] L. Adzhemyan, D. Evdokimov, M. Hnatič, E. Ivanova, M. Kompaniets, A. Kudlis, and D. Zakharov, "The dynamic critical exponent z for 2d and 3d ising models from five-loop expansion", Physics Letters A **425**, 127870 (2022).

[4] L. P. Kadanoff, "Scaling laws for ising models near T_c", Physics Physique Fizika **2**, 263–272 (1966).

[5] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications", Biometrika **57**, 97–109 (1970).

[6] B. Efron and R. Tibshirani, "Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy", Statistical Science **1**, 54–75 (1986).

[7] *Github repository physics760/final_project*, http://github.com/smilex555/physics760/tree/main/final_project, Accessed: 2023-02-26.

References II

[8] B. Nickel, "On the singularity structure of the 2D ising model susceptibility", J. Phys. A Math. Gen. **32**, 3889–3906 (1999).

[9] C. Duclut and B. Delamotte, "Frequency regulators for the nonperturbative renormalization group: a general study and the model a as a benchmark", 10.48550/ARXIV.1611.07301 (2016).