Licentiate Thesis

Examining Hausdorff dimension and Scaling behaviour with Worm algorithm

Simon Rydell

Theoretical Particle Physics, Department of Theoretical Physics,
School of Engineering Sciences
Royal Institute of Technology, SE-106 91 Stockholm, Sweden

Stockholm, Sweden 2018

Typeset in LaTeX

Akademisk avhandling för avläggande av teknologie licentiatexamen (TeknL) inom ämnesområdet teoretisk fysik.

Scientific thesis for the degree of Licentiate of Engineering (Lic Eng) in the subject area of Theoretical physics.

TRITA-FYS-2010:05

Printed in Sweden by Universitetsservice US AB, Stockholm May 2018

Abstract

Preface

Contents

Part I

Introduction and background material

Chapter 1

Introduction

Everyone agrees that the dimension of a point is zero, and that of a smooth line is one, but what about a set of points? A definition could be to say that the dimension is the minimum number of coordinates needed to describe every point in the set. Effectively, a point would describe itself, and a curve could be parametrized to the distance of some point on the same curve.

The situation is a bit more complex when examining fractals. Take for example the Koch curve, it starts out as a line segment of length L_0, and successively adds a 'bump', making the total length $L_1 = 4/3 \cdot L_0$. Iterating n times gives a line length of $L_n = (4/3)^n \cdot L_0$, and so the final fractal length is infinite.

Any two point on the final curve has a distance of infinity between them, so parametrization is impossible. But the area is still finite, so the dimension should intuitively be somewhere between one and two.

A useful concept here is the similarity dimension, defined by the scaling of each iteration. If m is the number of similar elements after an iteration and r is the scaling factor, the dimension is defined by $m = r^d$, or equivalently

$$d = \frac{\ln m}{\ln r} \tag{1.1}$$

So for the Koch curve, each segment is divided into fourths with each having one third the length from the previous iteration, giving it a dimension of $\ln 4 / \ln 3 \approx 1.26$.

Chapter 2

Background

2.1 Ising model

The Ising model consists of discrete atomic spins S that can be found in two states represented by the values $\{-1, 1\}$. This can be applied to a lattice where S_i is the spin of lattice site i.

The energy of such a configuration is then given by the Hamiltonian

$$H = -J \sum_{\langle ij \rangle} S_i S_j \tag{2.1}$$

Where J is the bond strength in the lattice.

2.1.1 Loop expansion

Let $K = \beta J$ where $\beta = 1/k_{\mathrm{B}}T$. Then from Equation (2.1)

$$\beta E = -K \sum_{\langle ij \rangle} S_i S_j \tag{2.2}$$

The partition function Z can therefore be written as

$$Z = \sum_{\text{all states}} e^{-\beta E} = \sum_{\text{all states}} e^{K \sum_{\langle ij \rangle} S_i S_j} = \sum_{\text{all states}} \Pi_{\langle ij \rangle} e^{K S_i S_j} \tag{2.3}$$

Since $S_i S_j = \pm 1$ Euler identities can be used to expand the exponential in Equation (2.3).

$$\begin{aligned} e^{KS_iS_j} &= \frac{e^K + e^{-K}}{2} + S_iS_j\frac{e^K - e^{-K}}{2} \\ &= \cosh(K) + S_iS_j\sinh(K) \\ &= \{T = \tanh(K)\} \\ &= (1 + TS_iS_j)\cosh(K) \end{aligned}$$

In a $2D$ lattice with N spins and periodic boundary conditions there are $2N$ bonds. Therefore the partition function is

$$\begin{aligned} Z &= \sum_{\text{all states}} \Pi_{\langle ij\rangle}(1 + TS_iS_j)\cosh(K) \\ &= \cosh^{2N}(K)\cdot 2^N\left(2^{-N}\sum_{\text{all states}}\Pi_{\langle ij\rangle}(1 + TS_iS_j)\right) \\ &= \cosh^{2N}(K)\cdot 2^N Z' \end{aligned}$$

Where

$$\begin{aligned} Z' &= 2^{-N}\sum_{\text{all states}}\Pi_{\langle ij\rangle}(1 + TS_iS_j) \\ &= 2^{-N}\sum_{S_1=\pm1}\sum_{S_2=\pm1}\cdots\sum_{S_N=\pm1}\left(1 + T\sum_{l=1}S_iS_j + T^2\sum_{l=2}(S_iS_j)(S_{i'}S_{j'}) + \ldots\right) \end{aligned}$$

Where the sums $\sum_{l=L}$ should be interpreted as the sum over all sets where the link length is L. Link length is the number of coupling terms S_iS_j as can be seen in Figure (2.1).

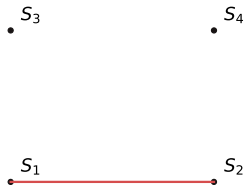
Since $\sum_{S_i=\pm1} S_i = 0$, only terms with an even number of S_i are contributing to Z'. Call these terms closed, indicating that they represent a closed loop. The sum over all contributing terms gives a factor of 2^N.

Rewriting Z' in terms of loop lengths gives

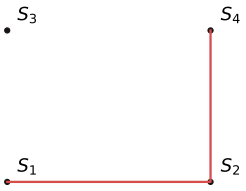
$$Z' = \sum_L g(L)T^L \tag{2.4}$$

Where $g(L)$ is the number of loops with length L. Finally, the partition function can be written as

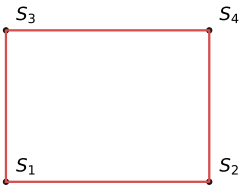
$$Z = 2^N\cosh^{2N}(K)\sum_L g(L)T^L \tag{2.5}$$



(a) $(S_1S_2),\ L = 1$



(b) $(S_1S_2)(S_2S_4),\ L = 2$



(c) $(S_1S_2)(S_2S_4)(S_4S_3)(S_3S_1),\ L = 4$

Figure 2.1: Link structure of an Ising lattice where (a) and (b) are open while (c) is closed.

2.1.2 Correlation Function

By the fluctuation-dissipation theorem the susceptibility can be written as [1]

$$\chi = \frac{\beta}{N} \sum_{ij} G_{ij} \tag{2.6}$$

Where $G_{ij} = \langle S_i S_j \rangle - \langle S_i \rangle^2$ is the connected correlation function between two lattice points i and j.

In the high-temperature expansion the term $\langle S_i \rangle^2$ goes to zero. So for a $2D$ Ising model

$$G_{ij} = \frac{1}{Z} \sum_{\text{all states}} S_i S_j e^{-\beta E} \tag{2.7}$$

$$= \{\text{See Section 2.1.1}\} \tag{2.8}$$

$$= \frac{1}{Z} \cosh^{2N}(K) \sum_{\text{all states}} S_i S_j \Pi_{\langle ij \rangle} (1 + \tanh(K) S_i S_j) \tag{2.9}$$

So the acceptance probability (See Section 3.4) is [2]

$$A_{ab} = \begin{cases} \min\left(1, \tanh(K)\right), & \text{To create a link between } a \text{ and } b. \\ \min\left(1, \frac{1}{\tanh(K)}\right), & \text{To remove a link between } a \text{ and } b. \end{cases} \tag{2.10}$$

2.2 *n*-vector model

The n-vector model describes a classical system of n-dimensional classical spins s_i of unit length interacting on a lattice. It is a generalization of the Ising model where each spin can have a continuous set of values. The Hamiltonian is then

$$H = -J \sum_{\langle ij \rangle} s_i \cdot s_j \tag{2.11}$$

where J is the bond strength and $\langle ij \rangle$ refers to a nearest neighbour interaction.

2.3 XY model

A special case of the n-vector model is the XY model when $n = 2$. Here the spins are two dimensional rotors as $s_i = (\cos\theta_i,\ \sin\theta_i)$. This yields the Hamiltonian

$$H = -J \sum_{\langle ij \rangle} \cos(\theta_i - \theta_j) \tag{2.12}$$

The partition function is therefore

$$Z = \prod_i \int \frac{\mathrm{d}\theta_i}{2\pi} e^{-\beta H} = \prod_i \int \frac{\mathrm{d}\theta_i}{2\pi} e^{K \sum_{\langle ij \rangle} \cos(\theta_i - \theta_j)} \tag{2.13}$$

where $K = J\beta$.

2.3.1 Loop expansion

Since equation (2.13) is invariant under the transformation $\theta_i - \theta_j \to \theta_i - \theta_j + 2\pi n$, $n \in \mathbb{Z}$, it can be expanded using the identity

$$e^{\alpha \cos\beta} = \sum_{\gamma=-\infty}^{\infty} I_\gamma(\alpha) e^{i\gamma\beta} \tag{2.14}$$

where $I_\gamma(\alpha)$ is the modified Bessel function. Using that $e^{\sum_i x_i} = \prod_i e^{x_i}$

$$Z = \prod_i \int \frac{\mathrm{d}\theta_i}{2\pi} \sum_{J_{\langle ij\rangle}=-\infty}^{\infty} \prod_{b=\langle ij\rangle} I_{J_{\langle ij\rangle}}(K) e^{iJ_{\langle ij\rangle}(\theta_i-\theta_j)} \tag{2.15}$$

$$= \prod_i \sum_{J_b} \left(\int \frac{\mathrm{d}\theta_i}{2\pi} e^{iN_i(\theta_i-\theta_j)} \right) \left(\prod_b I_{J_b} \right) \tag{2.16}$$

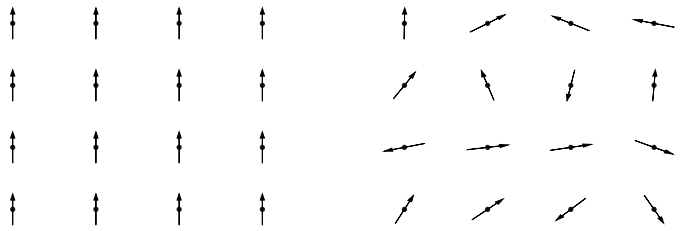
Where in the last step, $\prod_{\langle ij\rangle} e^{iJ_{\langle ij\rangle}(\theta_i-\theta_j)} = e^{iN_i(\theta_i-\theta_j)}$. N_i is therefore the sum of J for the nearest neighbours of site i. Noting that

$$\int \frac{\mathrm{d}\theta_i}{2\pi} e^{iN_i(\theta_i-\theta_j)} = C\delta_{N_i 0} \tag{2.17}$$

leads to the conclusion that the sum of incoming and outgoing flux J into a site i must be zero, in other words, the configurations are divergence free. This in turn means that a configuration of the system must contain closed loops of flux J.

2.3.2 Winding number

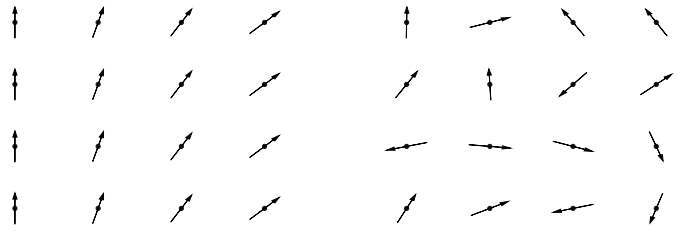
In the ground state all the spins are aligned, while at higher energy states, the spins are pointed in random directions as can be seen in Figure (2.2).



(a) Ground state (b) Higher energy state

Figure 2.2: Energy states for XY model

Therefore, making a constant phase shift $\Phi_\mu = \frac{A}{L}$ of $\theta_i - \theta_j$ in the μ direction would change the energy drastically for the ground state while, on a statistical average, not change the higher states energy at all (see Figure (2.3)).



(a) Ground state (b) Higher energy state

Figure 2.3: A phase shift for $\mu = x$

The free energy change ΔF for such a shift is

$$\Delta F = L^d \cdot \frac{1}{2}\rho_s \left(\frac{A}{L}\right)^2 \Rightarrow \rho_s = \lim_{A\to 0} L^{2-d}\frac{\partial^2 \Delta F}{\partial A^2} \tag{2.18}$$

where d is the dimension and ρ_s is the superfluid density which is zero for a high energy state. The free energy is

$$F = -T\ln(Z) \Rightarrow F'' = T\left(\left(\frac{Z'}{Z}\right) - \left(\frac{Z''}{Z}\right)^2\right) \tag{2.19}$$

where $F' = \partial F/\partial A$. Examining Z from (2.16) with the added shift

$$Z = \prod_i \int \frac{\mathrm{d}\theta_i}{2\pi} \sum_{J_{\langle ij\rangle}=-\infty}^{\infty} \prod_{b=\langle ij\rangle} I_{J_{\langle ij\rangle}}(K) e^{iJ_{\langle ij\rangle}(\theta_i-\theta_j+\Phi_\mu)} \tag{2.20}$$

$$= \prod_i \sum_{J_b} \left(\int \frac{\mathrm{d}\theta_i}{2\pi} e^{iN_i(\theta_i-\theta_j)}\right)\left(\prod_b I_{J_b}\right) \cdot e^{iA\frac{1}{L}\sum_i J_{i,i+\mu}} \tag{2.21}$$

$$\tag{2.22}$$

where in the last step

$$\prod_i \left(\prod_{\langle ij \rangle} e^{iJ_{\langle ij \rangle} \Phi_\mu} \right) = \tag{2.23}$$

$$\{\Phi_\mu \neq 0 \text{ only for neighbours in the } \mu \text{ direction}\} = \tag{2.24}$$

$$\prod_i \left(e^{iJ_{i,i+\mu} \Phi_\mu} \right) = \tag{2.25}$$

$$e^{iA \frac{1}{L} \sum_i J_{i,i+\mu}} \tag{2.26}$$

Introduce the winding number in the μ direction as

$$W_\mu = \frac{1}{L} \sum_i J_{i,i+\mu} \tag{2.27}$$

Intuitively, this describes the net flux in the μ-direction. Given a loop within the bounds of the lattice, the winding number is always zero. This is since an equal amount of flux in $+\mu$ as in $-\mu$ is needed to form a loop. However, this is not the case for a percolating cluster going, for example, from $-\mu$ to $+\mu$ connecting with periodic boundary conditions. For such a 'winding' cluster, the winding number will be $+1$. An example can be seen in figure (2.4).

Using the definition (2.27) for the winding number in the partition function in equation (2.22) yields

$$Z = \sum_{J_b} \left(\prod_b I_{J_b} \right) \prod_i \left(\int \frac{\mathrm{d}\theta_i}{2\pi} e^{iN_i(\theta_i - \theta_j)} \right) \cdot e^{iAW_\mu} \tag{2.28}$$

$$= \sum_{J_b,\ N_i = 0} Z_0 \cdot e^{iAW_\mu} \tag{2.29}$$

$$= \sum_{W_\mu} Z_0 \cdot e^{iAW_\mu} \tag{2.30}$$

Using this result in equation (2.19)

$$\frac{\partial^2 F}{\partial A^2} = T \left(\left(\frac{\sum_{W_\mu} (iW_\mu) Z_0 e^{iAW_\mu}}{\sum_{W_\mu} Z_0 e^{iAW_\mu}} \right)^2 - \frac{\sum_{W_\mu} (-W_\mu^2) Z_0 e^{iAW_\mu}}{\sum_{W_\mu} Z_0 e^{iAW_\mu}} \right) \tag{2.31}$$

$$= T \left(-\langle W_\mu \rangle^2 + \langle W_\mu^2 \rangle \right) \tag{2.32}$$

$$= T \langle W_\mu^2 \rangle \tag{2.33}$$

Where $\langle W_\mu \rangle = 0$ since there is an equal chance of percolating from $-\mu$ to μ as the other way around.

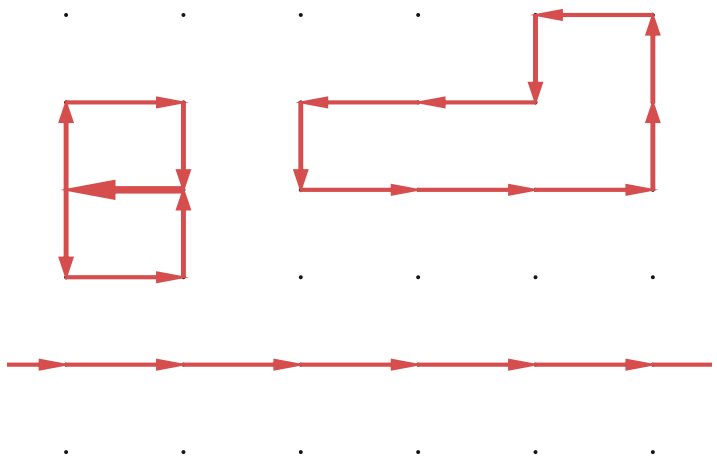


Figure 2.4: Three flux clusters on a square lattice. One percolating cluster with $W_x = +1$. The size of each arrow corresponds to the number of flux quanta between two sites.

The superfluid density can finally be determined as

$$\rho_s = L^{2-d} T \langle W_\mu^2 \rangle \tag{2.34}$$

2.3.3 Villain approximation

The Villain model the Hamiltonian has the form [3]

$$H = \sum_{ij} V_{ij}(\theta_i - \theta_j) + \sum_i U(\theta_i) \tag{2.35}$$

By taking

$$V_{ij}(\theta_i - \theta_j) = -J \cos(\theta_i - \theta_j) \tag{2.36}$$

$$U(\theta_i) = 0 \tag{2.37}$$

The XY model in Equation (2.12) is recovered.

The approximation then made by Villain was to replace the magnetic Hamiltonian by an approximate one [3].

Through a series of transformations the energy in this approximation can be written as [4]

$$E = \frac{1}{2}\sum_i J_i^2 \tag{2.38}$$

Where J_i^2 is the flux from site i.
The acceptance probability (See Section 3.4) is therefore

$$A_{ab} = \min\left(1, e^{-\Delta E}\right) \tag{2.39}$$

$$= \min\left(1, e^{-\frac{1}{2}\Delta J_{ab}^2}\right) \tag{2.40}$$

Where J_{ab} is the link between site a and b.

2.3.4 Energy Scaling

The scaling behaviour of the heat capacity is

$$C = \frac{e}{t} = \tilde{a}t^{-\alpha} + \tilde{b} \tag{2.41}$$

where $t = |T - T_c|$, $\alpha = -0.01$, and $\tilde{a},\ \tilde{b}$ are some constants.
Therefore, the energy per site, e is

$$e = at^{1-\alpha} + b \tag{2.42}$$

And the total energy

$$E = L^d(at^{1-\alpha} + b) \propto L^d, \text{ at } T = T_c \tag{2.43}$$

where d is the dimension of the sample. This scaling behaviour for the total energy can be seen in Figure (4.9) for $d = 3$.

2.4 Hausdorff dimension

Let X be a metric space, α be some positive real number, then the α-Hausdorff measure of a subset $A \subset X$ is defined as [5]

$$\mathcal{H}_\alpha^\delta(A) = \inf\left\{\sum_{B\in\mathcal{B}} (\text{diam}(B))^\alpha\right\} \tag{2.44}$$

Where $\mathcal{B}$ is a cover of A of closed balls with diameter no larger than δ.
Taking the limit where $\delta \to 0$, the number of possible covers decreases. Since the limit is bounded from below, the limit exists [6] and

$$\lim_{\delta \to 0} \mathcal{H}^{\delta}_{\alpha}(A) = \mathcal{H}_{\alpha}(A) \in [0, \infty) \tag{2.45}$$

Examining Equation (2.45) gives

$$\lim_{\delta \to 0} \mathcal{H}^{\delta}_{\alpha}(A) = \lim_{\delta \to 0} \inf \left\{ \sum_{B \in \mathcal{B}} (\text{diam}(B))^{\alpha} \right\} \tag{2.46}$$

$$\leq \lim_{\delta \to 0} \inf \left\{ \sum_{B \in \mathcal{B}} \delta^{\alpha} \right\} \tag{2.47}$$

$$= \lim_{\delta \to 0} \inf \left\{ N^{A}_{\delta} \delta^{\alpha} \right\} \tag{2.48}$$

Where N^{A}_{δ} is the number of balls with diameter δ that can cover A. Though not a proof, one can intuitively say that if for some $\alpha > 0$ the limit is finite, then $\mathcal{H}_{\alpha'} = 0$ for each $\alpha' > \alpha$. Therefore the number

$$\dim_H(A) = \inf\{\alpha > 0 : \mathcal{H}_{\alpha}(A) = 0\} \tag{2.49}$$

exists and is called the Hausdorff dimension of A [5].

2.5 Box dimension

Given some subset A of a metric space X, let $N(\epsilon)$ be the number of boxes with side length ϵ needed to cover A. Then the box dimension d of A is defined as [7]

$$d = \lim_{\epsilon \to 0} \frac{\ln N(\epsilon)}{\ln 1/\epsilon} \tag{2.50}$$

If the limit exists.

For intuition it helps to look at some examples. If A is a smooth line of length l, then the number of boxes needed to cover A scales as

$$N_l(\epsilon) \propto \frac{L}{\epsilon} \tag{2.51}$$

While for some two dimensional region with area Λ the boxes needed is

$$N_{\Lambda}(\epsilon) \propto \frac{\Lambda}{\epsilon^2} \tag{2.52}$$

Such that for a d-dimensional subset A, the boxes needed will scale as

$$N(\epsilon) \propto \frac{1}{\epsilon^d} \Rightarrow d = \frac{\ln N(\epsilon)}{\ln 1/\epsilon} \tag{2.53}$$

Note that the box dimension and the Hausdorff dimension coincides for fractals that satisfy the open set condition [8].

The open set condition says that for a sequence of contractions $c_1, c_2, ..., c_m$ exists a nonempty open set V such that [9]

$$\cup_{i=1}^{m} c_i(V) \subset V, \text{ and } c_i(V) \cap c_j(V) = \emptyset \text{ for } i \neq j \tag{2.54}$$

Intuitively, this means that the images $c_i(V)$ do not overlap 'too much'.

2.5.1 Scaling Dimension

Another way of determining the Hausdorff dimension is to consider a scaling system [10]. The scaling behaviour of the largest cluster should then follow

$$N = L^{D_H} \tag{2.55}$$

Where N is the number of links in the largest cluster, L is the linear system length and D_H is the Hausdorff dimension.

This scaling relation can be seen in Figure (4.3).

Chapter 3

Method

3.1 Monte Carlo Simulations

A Monte Carlo simulation is a stochastic algorithm to get statistical results from some system where there are many degrees of freedom.

Markov Chain Monte Carlo is the idea to use Markov Chains to propose the next step in the algorithm. Using a well-chosen stationary probability distribution, together with ergodicity (Section 3.2), the desired distribution will be sampled.

3.2 Ergodicity

An ergodic system is one where its time average coincides with its ensemble average.

Intuitively, ergodicity is the assumption that a Markov chain starting from some state S_a with a non zero Boltzmann weight can reach any other state S_b within a finite number of updates [11].

This is necessary to assume, since otherwise there could be a non zero contribution to the partition function not being sampled by the Markov chain.

3.3 Detailed Balance

Generally, a Markov process can be described through the Master equation

$$\frac{\mathrm{d}P_a}{\mathrm{d}t} = \sum_{a \neq b} (P_b W_{ba} - P_a W_{ab}) \tag{3.1}$$

where P_a is the probability to find the system in the state a, and W_{ab} is the transition rate from the state a to b. This equation describes the equilibrating of P_a into $P_a^{eq} \propto e^{-\beta E_a}$. The resulting equation is called detailed balance

$$W_{ba}e^{\beta E_b} = W_{ab}e^{\beta E_a} \tag{3.2}$$

This describes an equal rate of flow into the state a as out of it.
Together with ergodicity, detailed balance ensures a correct algorithm [2].

3.4 Metropolis Algorithm

Splitting the transition rate as

$$W_{ab} = T_{ab}A_{ab} \tag{3.3}$$

where T_{ab} is a trial proposition probability and A_{ab} is an acceptance probability. Letting $T_{ab} = T_{ba},\ \forall a,\ b$ and choosing one of the acceptance probabilities $A_{ab},\ A_{ba}$ to be equal to 1, together with detailed balance, the Metropolis algorithm is realized [2]. The acceptance ration is therefore

$$A_{ab} = \min\left\{1, \frac{P_b}{P_a}\right\} = \min\{1, e^{-\beta(E_b - E_a)}\} \tag{3.4}$$

3.5 Worm Algorithm

The Worm algorithm operates on graph configurations instead of individual spins. This way, the algorithm can stay local and can avoid, to some extent, the so called critical slowdown that happens near transition points [12].

The algorithm still uses the Metropolis acceptance rates and therefore it fulfills detailed balance (see Section 3.3).

Intuitively, the 'worm' part can be seen as a 'magic marker' that can connect or remove the connection between two sites. This way the marker draws patterns onto a lattice that corresponds to a configuration, typically of the partition function. This paper is only concerned with whenever the marker reaches the same lattice site that it started on, forming loops, or clusters.

3.6 Hoshen Kopelman

To find the clusters a modified version of the Hoshen Kopelman algorithm was used. A raster scan is used to label disjoint sets into groups with some canonical label [13]. It is a variant on the union-find algorithm and is most easily described through the associated functions. Intuitively, applying the find function on a site i returns the canonical, often implemented as the smallest, label in the cluster that i belongs to. Union uses find to ensure that two sites i and j are connected by setting the canonical label of i to that of j (or vice versa).

An example implementation would be to have a 2D graph without periodic boundary conditions of zeros and ones, where a site is occupied if it has a one

associated with it, and unoccupied otherwise. A disjoint set here is a number of occupied sites neighbouring each other with unoccupied sites surrounding them. For simplicity the scan can start in the lower left corner, moving right and up, while search for neighbours left and down, ensuring that if a neighbouring site is occupied, it has been labeled before.

Start by setting each site to a unique label, putting all sites in individual clusters. Go through the lattice until an occupied site i is found. Search the neighbours below and to the left. If none of these neighbours are occupied, label i have a unique label and move to the next site. If i has one occupied neighbour it must have been labeled before, so i inherits the neighbours label. Finally if both neighbours are occupied, site i must be connecting a cluster and a union is performed on the neighbours to join their labels. A final pass through the lattice using the find function ensures that all sites have their canonical label.

In this paper an occupied site corresponds to a site with connections to the neighbouring sites (in the 2D example above, each site could have four such connections). In the original paper by Hoshen and Kopelman the labels for the sites who did not originally carry the canonical label, were set to a negative integer, symbolizing that they were aliases. A positive value was used at the canonical label, showing the number of sites in that cluster. This was not used in this project since the number of links in a cluster is not necessarily equal to the number of sites.

3.7 Graph Dividing Algorithm

In order to calculate the box dimension the lattice need to be divided into boxes of decreasing size. A step by step instruction of a graph dividing algorithm is provided below, and an implementation in pseudocode is available in the Appendix at Section **??**.

For brevity some abbreviations are introduced.

d = dimension	l_i = side length of the current box
l_0 = side length of the smallest box allowed	e_i^j = vector of length $l_i/2$ in the j'th direction
perm(v) = All permutations of v	s_i = starting site of the current box

1. If $l_i \geq l_0$, go to 2, else stop.

2. Save all sites in the current box, starting from s_i going l_i in d directions.

3. Find all starting points for new boxes.

   (i) Form the matrix $E = (e_i^0, e_i^1, \ \ldots, e_i^d)^T$

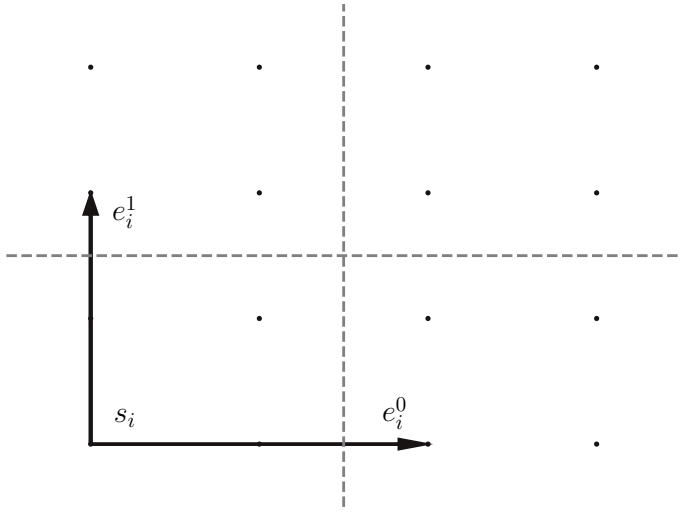


Figure 3.1: One step in the graph dividing algorithm where $l_i = 4$. e_i^0 and e_i^1 are drawn from site s_i. Summed permutations of $\{e_i^0, e_i^1\}$ give the starts for the next boxes. The next iteration of boxes are shown via the dividing dotted lines.

(ii) For all vectors v_k in perm$(0, 0, \ldots, 0)$, perm$(1, 0, \ldots, 0)$, $\ldots$, perm$(1, 1, \ldots, 1)$, create the new start s_k as

$$s_k = v_k E$$

4. For each start s_k:

   (i) $s_i = s_k$, $l_i = l_i/2$

   (ii) Go to 1.

This algorithm can be written to perform in linear time, as can be seen in Figure (3.2).

3.8 Error Estimation

In this thesis a number of error estimation techniques were used to know how much data was needed for each measurement. In this section the techniques will be described intuitively.

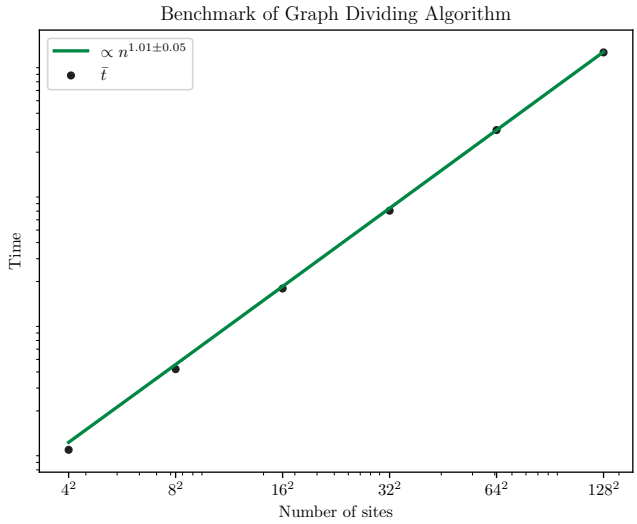


Figure 3.2: Loglog plot of a benchmark of the graph dividing algorithm. The y-axis show the time taken to perform one full graph divide normalized against the smallest time value.

3.8.1 Monte Carlo Error Estimation

Given a Monte Carlo simulation where polling of some quantity A has been done N times an estimation of the expectation value of A is

$$\bar{A} = \frac{1}{N}\sum_{i=1}^{N} A_i \tag{3.5}$$

where each sampling was labeled A_i. To show that this is an unbiased estimator the expectation value of the difference between the estimation and the real value $\langle A \rangle$ is used.

$$\langle \bar{A} - \langle A \rangle \rangle = \langle \bar{A} \rangle - \langle A \rangle \tag{3.6}$$
$$= \left\langle \frac{1}{N} \sum_{i=1}^{N} A_i \right\rangle - \langle A \rangle \tag{3.7}$$
$$= \frac{1}{N} \sum_{i=1}^{N} \langle A_i \rangle - \langle A \rangle \tag{3.8}$$
$$= \frac{1}{N} \sum_{i=1}^{N} \langle A \rangle - \langle A \rangle \tag{3.9}$$
$$= \frac{1}{N} N \langle A \rangle - \langle A \rangle = 0 \tag{3.10}$$

where the fact that A_i is a random sampling from the distribution of A was used in (3.9).

The standard deviation of this estimate can be calculated through the variance.

$$\sigma_{\bar{A}}^2 = V\left(\bar{A} - \langle A \rangle\right) \tag{3.11}$$
$$= V\left(\bar{A}\right) - V\left(\langle A \rangle\right) \tag{3.12}$$
$$= \{\langle A \rangle \text{ is a constant} \Rightarrow V(\langle A \rangle) = 0\} \tag{3.13}$$
$$= V\left(\frac{1}{N} \sum_{i=1}^{N} A_i\right) \tag{3.14}$$
$$= \{\text{Monte Carlo simulations give independent samples}\} \tag{3.15}$$
$$= \frac{1}{N^2} \sum_{i=1}^{N} V(A_i) \tag{3.16}$$
$$= \{V(A_i) = \sigma_A^2\} \tag{3.17}$$
$$= \frac{1}{N^2} N \sigma_A^2 = \frac{\sigma_A^2}{N} \tag{3.18}$$

or

$$\sigma_{\bar{A}} = \frac{\sigma_A}{\sqrt{N}} \tag{3.19}$$

So the standard error in this estimation decreases as $N^{-1/2}$.

3.8.2 Bootstrap

Bootstrap is a resampling method to examine a probability distribution. In this thesis it was used to estimate the error propagation of parameters in curve fitting.

Given a set $\boldsymbol{x}$ of N measurements from an unknown distribution $\hat{\phi}$, some statistical calculation of interest can be done as $\theta = s(\boldsymbol{x})$. A resampling $\boldsymbol{x}_0$ of $\boldsymbol{x}$ comprised of N random measurements from $\boldsymbol{x}$ (where one measurement can be included several times), can then be used to calculate $\theta_0^* = s(\boldsymbol{x}_0)$. Repeating this N_B times gives an estimate $\theta^* = (\theta_0^*, \theta_1^*, ..., \theta_{N_B}^*)$ of the distribution $\hat{\theta}$. Assuming N_B is large then, by the central limit theorem, $\hat{\theta}$ is a normal distribution with some standard deviation σ_θ that can be used as an error estimation for θ.

3.9 Optimization

A large part of this project was spent optimizing the simulations. To fascilitate this process, the Google C++ framework *benchmark* was used for measuring and comparing running times of different algorithms. Only the C++ code was optimized in this way since the prototypes written in Python were not concerned with computational speed. What follows is a summation of the biggest optimizations that were achieved during this project.

3.9.1 Lattice implementation - Squashing the Graph

During most of the calculations a sweep through the entire lattice was necessary. The computational time taken by such a sweep is heavily affected by the implementation of the lattice data structure.

The first implementation used an array of arrays to represent the lattice. This has the advantage of a similar interface to mathematical matrices with rows and columns. However, it is much slower than a single array, or a squashed graph, most likely due to cache misses[14]. Figure (3.3) shows a comparison between the time it takes to do one full sweep in the two implementations.

Furthermore, this simplifies the generalization to lattices of different dimensions since the need to allocate a set number of arrays in each array disappears. It also makes it easier to reuse the same algorithm for lattices of different dimensions.

The mapping used between the array index and the Euclidean space is

$$n = x + yL + zL^2 + ... \tag{3.20}$$

where L is the linear system size.

3.9.2 Saving Warmed Up Graphs

Since there is no reliable way to guess a physical state at the start of a simulation, each system must be 'warmed up' to produce valid results. One way of doing this is to continuously measure the quantity of interest, and while each measurement is quantifiably different from the previous one, keep updating the system.

When the system finally produces sufficiently close results, a representation of the graph can be stored on disk. In this project, the graph was saved to a simple

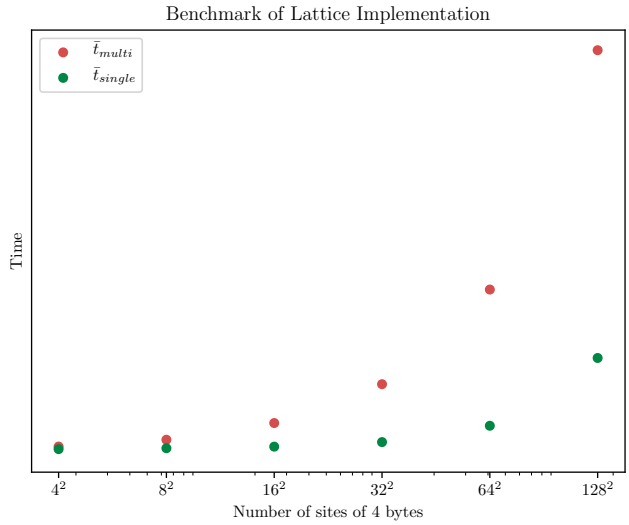


Figure 3.3: Plot of a sweep benchmark of two different lattice implementations. The red dots labeled $\bar{t}_{multi}$ is an array of arrays, while the green dots labeled $\bar{t}_{single}$ is a single array. The y-axis show the time taken to perform one full graph divide normalized against the maximum of the smallest time value for the two implementations.

text file that could be parsed using regex. This greatly helped with performance, and in some cases halved the simulation times.

The drawback of this solution is the need for a self-written parser. Since the simulation and plotting were done in two different programming languages, the better solution would be a combined interface such as an *sqllite* database.

3.10 Testing

Testing is an essential part of writing a simulation. The correctness of the code ensures that the physical model is aptly described. To facilitate the development, a number of coding practices were applied, such as unit testing (Section 3.10.1) and regression testing (Section 3.10.2). The tests for the prototypes were written in the Python standard library utility *unittest*, and those for the main simulation used the C++ framework *Catch2*.

3.10.1 Unit Testing

Unit testing refers to the practice of isolating a 'unit' of code and testing its correctness. A unit can be any small piece of code with an expected behaviour.

This was done by manually calculating the expected output of some code, given some input, and ensuring that the piece of code produced an equivalent result.

The parts of the simulation using a pseudorandom number generator were tested by randomly selecting a set of seeds on which the tests were run upon. The correctness is then assumed from this subset of possible inputs.

Combining multiple units into one test is called integration testing. This assumes that each unit is correct by itself, and it was used to more closely ressemble the actual simulation.

3.10.2 Regression Testing

Rerunning the relevant tests in a continuous manner after each update is called regression testing. This ensures that, however many unintended consequences were introduced during the update, the expected behaviour of the program is still intact.

This was done by writing a 'hook' such that, whenever a piece of code were recompiled, the relevant tests were subsequently compiled and run. With sufficient tests in place, the correctness of the code after the update was assumed.

Chapter 4

Results

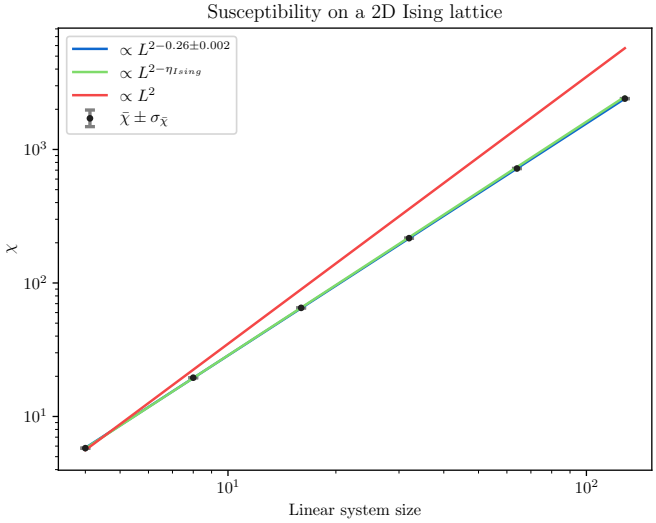


Figure 4.1: Scaling of susceptibility at T_c on an Ising lattice of varying sizes. The measured critical exponent $\eta = 0.26 \pm 0.002$ is compared to the theoretical value $\eta_{Ising} = 0.25$ [15]. The red line labeled $\propto L^2$ illustrates data where susceptibility scales with system size.

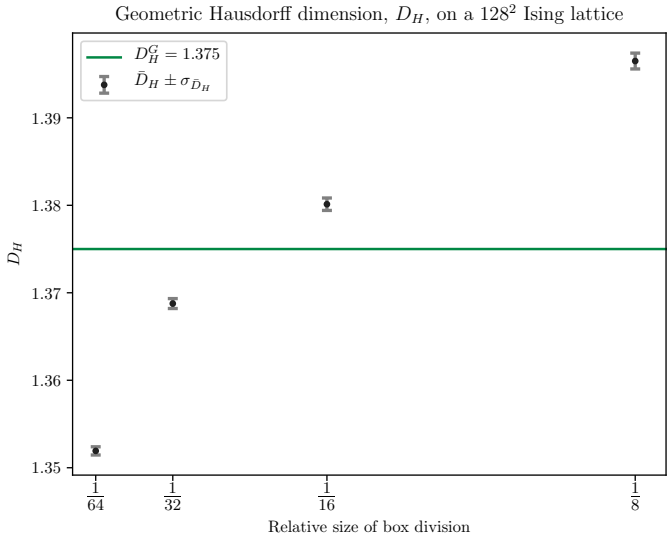


Figure 4.2: Hausdorff dimension of the maximum loop length on a 128^2 Ising lattice at T_c using the box dimension. The x axis shows the size of one box relative to the side length of the lattice. The green line indicates the theoretical dimension of the geometric Ising cluster, $D_H^G = 1.375$ [16]. Comparing to the smallest box size as $D_H = 1.35193 \pm 5 \cdot 10^{-4}$.

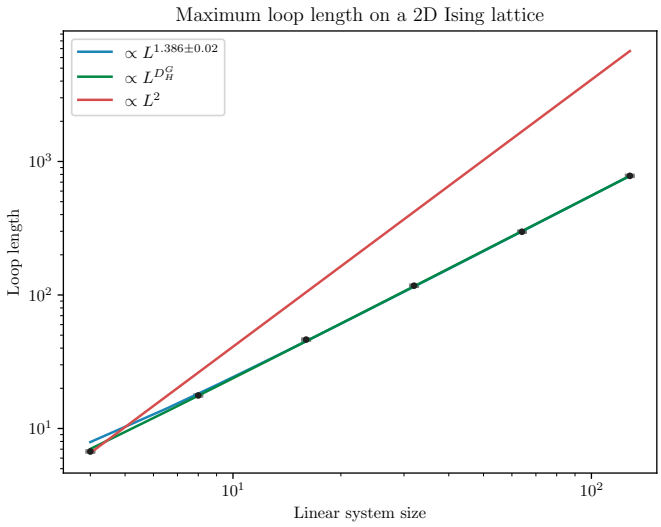


Figure 4.3: Log-log plot of the maximum loop length at T_c on an Ising lattice of varying sizes. The measured scaling factor is 1.38 ± 0.02 compared to the theoretical Hausdorff dimension, $D_H^G = 1.375$ [16]. The red line labeled $\propto L^2$ illustrates data where maximum loop length scales with system size.

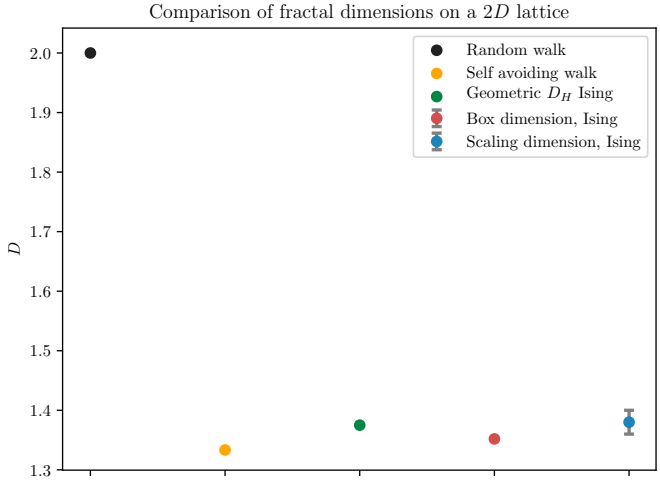


Figure 4.4: Comparison between different algorithms and ways of calculating the fractal dimension on a $2D$ lattice. The random walk has a dimension of 2, the self avoiding walk has $4/3$ [20]. The geometric Hausdorff dimension has an exact value of 1.375 [16]. This is then approximated using the worm algorithm and calculated using both the box counting method and the scaling dimension method explained in Section 2.5.1.

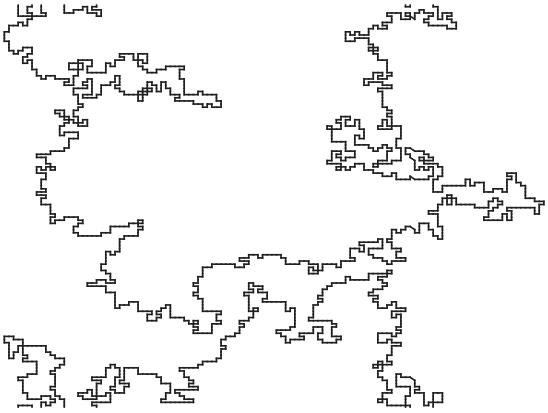


Figure 4.5: Isolated largest cluster on a 128^2 Ising lattice with periodic boundary conditions.

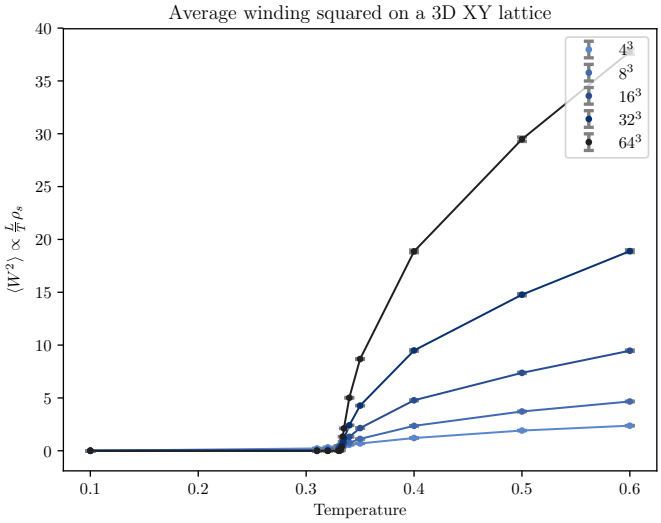


Figure 4.6: Average winding number squared, $\langle W^2 \rangle \propto \rho_s$, plotted on a 3D XY lattice of varying sizes. The overall structure of the average winding number squared as a function of the temperature is shown.

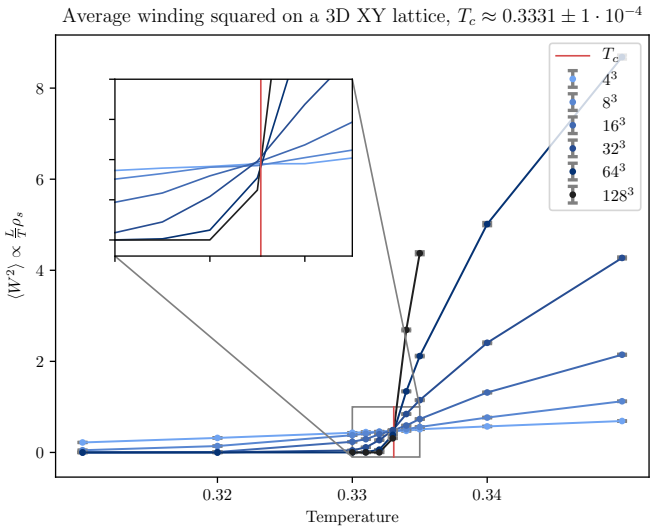


Figure 4.7: Average winding number squared, $\langle W^2 \rangle \propto \rho_s$, plotted on a 3D XY lattice of varying sizes. Due to the Villain approximation the transition is flipped such that $\rho_s \neq 0$ for $T > T_c$, and T_c is translated from ≈ 2.2 [17] to ≈ 0.333 indicated by the intersection. By taking the weighted average of the intersections the critical temperature can be estimated to $T_c = 0.3331 \pm 1 \cdot 10^{-4}$.

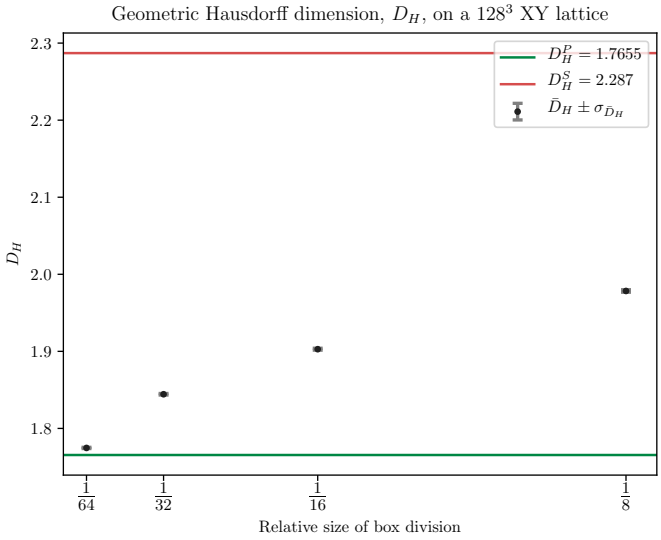


Figure 4.8: Hausdorff dimension of the maximum loop length on a 128^3 XY lattice at T_c using the box dimension. The x axis shows the size of one box relative to the side length of the lattice. Assuming that the smallest box size gives the best approximation, the result is $D_H = 1.77468 \pm 4 \cdot 10^{-6}$. This is compared to data from other papers $D_H^P = 1.7655 \pm 2 \cdot 10^{-3}$ [18] and $D_H^S = 2.287 \pm 2 \cdot 10^{-3}$ [19]

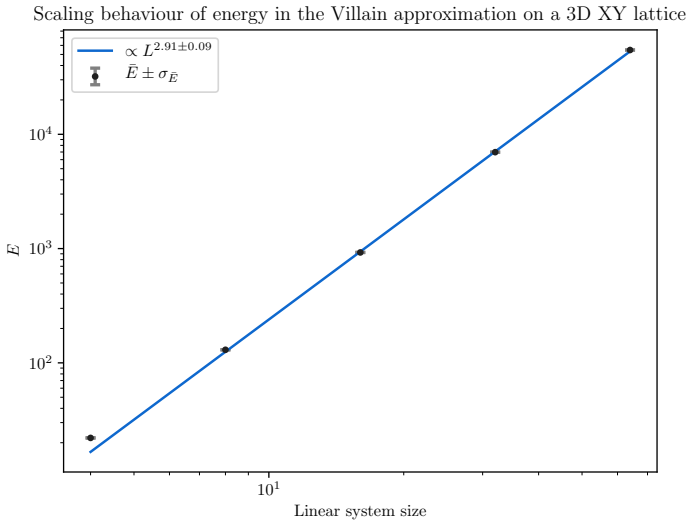


Figure 4.9: Log-log plot of the energy at T_c on an 3D XY lattice of varying sizes. The measured scaling factor is 2.91 ± 0.09. The energy in the Villain approximation is proportional to the sum of the squares of flux flowing through the lattice.

Chapter 5

Summary and discussion

5.1 Summary

The scaling exponent $\eta = 0.25$ of the susceptibility on a $2D$ Ising lattice was examined with the Worm algorithm. This gave good results of $\eta = 0.26 \pm 0.002$.

Two different ways of determining the Hausdorff dimension of largest clusters at T_c using the Worm algorithm was discussed. The scaling method examines the behaviour of the largest cluster as the system size increases. This gave the closest results to the geometrical Hausdorff dimension for a $2D$ Ising cluster of $D_H = 1.38 \pm 0.02$ compared to the analytical answer $D_H = 1.375$ [16].

The second method was to approximate the Hausdorff dimension using the box dimension. This gave close results, $D_H = 1.35193 \pm 5 \cdot 10^{-4}$, to the scaling method but seem to diverge from the correct result for small boxes.

The resulting Hausdorff dimension of the 2D Ising lattice lies between the fractal dimension of a self avoiding walk with $D_{SAW} = 4/3$ [20], and a pure random walk with $D_{RW} = 2$. This suggests a new type of self avoiding walk where 'twisted loops' (see Section 5.2) can occur. A full comparison can be seen in Figure (4.4).

The box counting method was then applied to a $3D$ XY lattice and gave the result $D_H = 1.77468 \pm 4 \cdot 10^{-6}$. This was then compared to previous results given by Hove, Mo and Sudbo as $D_H^S = 2.287 \pm 2 \cdot 10^{-3}$ [19], and the result from the article disputing that claim by Prokof'ev and Svistunov with the value $D_H^P = 1.7655 \pm 2 \cdot 10^{-3}$ [18]. This strengthens the claim by Pkof'ev and Svistunov.

Due to time constraints the scaling method was not applied to the $3D$ XY model.

To simplify the simulations, the Villain approximation was used (See Section 2.3.3). The resulting new critical temperature was calculated by measuring the winding number on a range of temperatures. Since the winding number is proportional to the superfluid density, a sharp decrease can be seen around the critical

temperature. By varying the system size, the intersections of all the drops can be taken as the critical temperature and was calculated to be $T_c = 0.3331 \pm 1 \cdot 10^{-4}$.

5.2 Discussion

The Worm algorithm is a good method for determining the Hausdorff dimension of a $2D$ Ising cluster. Scaling loop lengths gave the closest value $D_H = 1.38 \pm 0.02$ to the analytical answer $D_H = 1.375$[16]. This result could possibly be improved by more data as the analytical answer is within the error span.

Comparing the $2D$ Ising Hausdorff dimension to the fractal dimension of self avoiding walks, $D_{SAW} = 1.33$ [20], and random walks, $D_{RW} = 2$, the Ising Worm algorithm produces something in between. As can be seen in Figure (4.5) the cluster ressembles a self avoiding walk but with 'twisted loops' (See Figure (5.1)).

Figure 5.1: Examples of twisted loops.

This implies that this may be a new type of self avoiding walk that may have similar properties.

In all simulations trying to measure the length of a loop, the choice was made that whenever there is ambiguity of which path to follow, all paths were followed. This avoids the technique of choosing a random path used by other articles [19], and should in the average case return a longer loop. This can be intuitively understood by following the looping line in the number 8. Whenever the crossing in the middle is reached, one has to follow one of the branching paths. If the stop condition is whenever the starting position is reached, there is a chance of only including one of the loops. This was avoided by recursively following all paths until all loops are explored.

The Hausdorff dimension of the $3D$ XY lattice heavily favoured the result of Prokof'ev and Svistunov over Hove, Mo and Sudbo with this thesis producing $D_H = 1.77468 \pm 4 \cdot 10^{-6}$ compared to theirs $D_H = 1.7655 \pm 2 \cdot 10^{-3}$ [18]. The result

would have been strengthened by also comparing to the scaling method described in Section 2.5.1.

The winding number method provides a way of determining the critical temperature of a XY lattice. A weighted average of all intersections, weighted with the product of the linear system size of the two intersecting lines, was used to calculate the critical temperature. This method was chosen since the larger system was assumed to provide closer results to an infinite size system.

The choice to use the Hoshen Kopelman algorithm for cluster labeling (discussed in Section 3.6) reduced the simulation time considerably from the brute force methods used before. This together with a well chosen graph structure (discussed in Section 3.9.1) improved the data collection speed immensely.

All large scale simulations in this thesis were run on the Octopus cluster belonging to the department of theoretical physics at KTH.

Bibliography

[1] P. M. Chaikin and T. C. Lubensky, *Principles of condensed matter physics* (Cambridge University Press, 1995).

[2] J. C. Walter and G. Barkema, *An introduction to Monte Carlo methods*, Physica A: Statistical Mechanics and its Applications **418**, 78 (2015).

[3] J. Villain, *Theory of one- and two-dimensional magnets with an easy magnetization plane. II. The planar, classical, two-dimensional magnet*, Journal de Physique **36**, 581 (1975).

[4] J. V. José *et al.*, *Renormalization, vortices, and symmetry-breaking perturbations in the two-dimensional planar model*, Physical Review B **16**, 1217 (1977).

[5] J. Heinonen, *Lectures on Analysis on Metric Spaces* (Springer New York, 2001).

[6] W. Rudin, *Principles of Mathematical Analysis (McGraw Hill Publishing Company Ltd., 1964), ix 270 pp., 62s.*, Proceedings of the Edinburgh Mathematical Society **14**, 247 (1965).

[7] S. H. Strogatz, *Nonlinear Dynamics and Chaos With Applications to Physics, Chemistry, and Engineering*, 2 ed. (Westview Press Inc, Boulder, Colorado, 2014).

[8] K. Falconer, *Fractal Geometry* (John Wiley & Sons, Ltd, 2003).

[9] C. Bandt, N. V. Hung and H. Rao, *On the open set condition for self-similar fractals*, Proceedings of the American Mathematical Society **134**, 1369 (2005).

[10] M. Camarda *et al.*, *Methods to determine the Hausdorff dimension of vortex loops in the three-dimensionalXYmodel*, Physical Review B **74** (2006).

[11] H. C. Andersen and D. Chandler, *Robert W. Zwanzig: Formulated nonequilibrium statistical mechanics*, Proceedings of the National Academy of Sciences **111**, 11572 (2014).

[12] N. Prokofev and B. Svistunov, *Worm Algorithms for Classical Statistical Models*, Physical Review Letters **87** (2001).

[13] J. Hoshen and R. Kopelman, *Percolation and cluster distribution. I. Cluster multiple labeling technique and critical concentration algorithm*, Physical Review B **14**, 3438 (1976).

[14] P. J. Hanlon *et al.*, *The Combinatorics of Cache Misses during Matrix Multiplication*, Journal of Computer and System Sciences **63**, 80 (2001).

[15] M. Plischke and B. Bergersen, *Equilibrium Statistical Physics* (WORLD SCIENTIFIC, 2006).

[16] B. Duplantier, *Conformally Invariant Fractals and Potential Theory*, Physical Review Letters **84**, 1363 (2000).

[17] A. P. Gottlob and M. Hasenbusch, *Critical behaviour of the 3D XY-model: a Monte Carlo study*, Physica A: Statistical Mechanics and its Applications **201**, 593 (1993).

[18] N. Prokofev and B. Svistunov, *Comment on Hausdorff Dimension of Critical Fluctuations in Abelian Gauge Theories"*, Physical Review Letters **96** (2006).

[19] J. Hove, S. Mo and A. Sudbø, *Hausdorff Dimension of Critical Fluctuations in Abelian Gauge Theories*, Physical Review Letters **85**, 2368 (2000).

[20] T. Vilgis, *Flory theory of polymeric fractals - intersection, saturation and condensation*, Physica A: Statistical Mechanics and its Applications **153**, 341 (1988).