Universiteit Utrecht

Faculteit Bètawetenschappen

Master Thesis
Mathematical Sciences

Metastability in the Two Dimensional Long-Range Ising Model

Author:
Paul Horikx
4097556

Supervisor:
Dr. C. Spitoni
Utrecht University

Second Reader:
Dr. W. M. Ruszel
Utrecht University

November 27, 2020

Contents

1 Introduction

This thesis studies metastability in the Ising model in two dimensions. The Ising Model is lattice model to study the properties of ferromagnetic materials. We want to study metastable behaviour, a dynamical phenomenon of time scales where a system can seem at equilibrium on a short time scales, whereas it jumps between several regions of quasi-equilibrium on a longer time scale.

The Ising Model models ferromagnetism by looking at a lattice of spins of size $L \times L$ which are either $+1$ or -1. For a lattice Λ_L, our space of states is $\Omega_L := \{+1, -1\}^\Lambda$. For a given configuration $\sigma \in \Omega_L$, a Hamiltonian is calculated, allowing us to compare the energy of different configurations of spins on the model. This Hamiltonian consists of a term which models interactions between spins, $-\sum_{i,j\in\Lambda} J(i,j)\sigma_i\sigma_j$, and a general term which models an external field, $-h\sum_{i\in\Lambda}\sigma_i$.

The Hamiltonian for a state $\sigma \in \Omega_L$ is defined by

$$H_{\Lambda,h}(\sigma) = -\sum_{i,j\in\Lambda} J(i,j)\sigma_i\sigma_j - h\sum_{i\in\Lambda}\sigma_i, \tag{1}$$

where σ_i is the value of the spin at location i on the lattice Λ, J is a positive interaction term between i and j, and h is the value of an external field.

Generally, the interaction J is dependent only on the distance between i and j, so $J(i,j) = J(||i-j||)$.

In the standard Ising Model, the interaction between spins is simplified by only considering the interaction between direct neighbours. In this case, we get

$$J(||i-j||) = \begin{cases} 1 \text{ if } ||i-j|| = 1 \\ 0 \text{ otherwise.} \end{cases}$$

This would simplify the Hamiltonian to $H = -\sum_{i\in\Lambda}\sum_{j:||i-j||=1}\sigma_i\sigma_j - h\sum_{i\in\Lambda}\sigma_i$.

Building on a previous paper [8], we want to study a J which models long-range interactions. In this thesis we study J of the form

$$J = \frac{J}{||i-j||^\lambda} \tag{2}$$

This is the first study of the 2 dimensional long-range Ising model. By [8], "Long-range Ising models in low dimensions are known to behave like higher-dimensional short-range models". We want to study differences between the short-range 2 dimensional Ising model and the long-range 2 dimensional Ising model, particularly a difference in the paths the system is likely to take between different equilibria.

We will reference two states repeatedly in the thesis. We will denote these by $-\mathbb{1}, +\mathbb{1} \in \Omega_L$, where $-\mathbb{1}$ is the state where all spins are -1, and $+\mathbb{1}$ is the state where all spins are $+1$.

The positive component is the set of vertices in Λ for which the spin is $+1$.

1.1 Metropolis-Glauber Markov Chains

Metastability is a dynamical phenomenon. Therefore, we have to introduce dynamics on our system. We do this by defining a Metropolis-Glauber Markov chain on our system. This is a system also used in [5] and in [8]. We will formally define the Markov Chain in section 2.1.1, but we will introduce it informally here.

At every time-step, we select a random vertex to flip its spin. If this spin flip would lower the Hamiltonian, the spin on the vertex is always flipped. If the spin flip would increase the Hamiltonian, the spin is flipped with probability

$$e^{-\beta\Delta H}, \tag{3}$$

where ΔH is the difference in the Hamiltonian that flipping the spin makes, and so is always negative in this case. β is a variable which in statistical physics is inversely related to the temperature. We note that for each time-step, only a single spin is flipped. Therefore, the minimum time it would take the system to travel from a state η to a state η' which differs on the value of the spin on k vertices, is k time steps.

The Metropolis-Glauber weights are chosen such that the Gibbs measure is a reversible measure for this Markov Chain [5]. This measure is therefore invariant. In statistical physics, the Gibbs measure describes the canonical ensemble, a system in thermal equilibrium with a heat bath [2].

Since the system always flips the spin on the selected vertex if this lowers the Hamiltonian but flips it with a probability proportional to the increase of the Hamiltonian if it does not, the system will tend to follow the energy gradient.

We note that lower values of β increase the chaotic behaviour in the system. If flipping a spin results in a $\Delta H > 0$, the probability of flipping the spin under the dynamics increases with decreasing β.

1.2 Metastability

We want to study metastability in this model. A rigorous definition of metastability will come in the next chapter. Intuitively, metastability is the phenomenon where the system seems at equilibrium at shorter time scales, but jumps between several states of quasi-equilibrium at longer time scales. By [3], "at a short time-scale, the systems appears to be in an equilibrium state, but really explores only a confined section of its available phase space, while, at much larger time scales, it undergoes transitions between such metastable states."

We could understand a metastable state as being in a valley, whereas the stable state, the global minimum, is in the next valley over a mountain. The system initially only explores the valley in which the metastable lies, only when the system goes over the mountain it can go to the global stable state.

In our model, the state where all spins are positive is the metastable state for relevant choices of parameters $\lambda, h > 0$. We choose $h > 0$ so we get a difference in Hamiltonian between the state $-\mathbb{1}$ and $+\mathbb{1}$, where the state $+\mathbb{1}$ has a lower Hamiltonian. Therefore, the state $+\mathbb{1}$ is the global stable state, since it is the state with the lowest Hamiltonian. However, there is no path between them for which energy strictly decreases for each step. If we would choose $h = 0$, the Hamiltonian for $-\mathbb{1}$ and $+\mathbb{1}$ would be equal and therefore they would both be stable states.

We are interested in studying this phenomenon of metastability which occurs because of the specific Markov chain, and we want to look at its specifics for this model.

1.3 Goals

We want to see how the long-range interaction might change several metastability scenarios.

We are interested in the critical droplet. We have two approaches to study the size and shape of the critical droplet. The first is a direct comparison between several shapes, in which we determine which of the shapes is more efficient and at which size they reach their maximal Hamiltonian. We propose that the circle is a more efficient shape than the square for values of λ with significant long-range interaction.

We do not solve the variational problem of the most efficient path from $-\mathbb{1}$ to $+\mathbb{1}$. Instead, our goal is to compare two different mechanisms for the growth of the positive component.

The second approach to studying the critical droplet is a proposed algorithm to find the configuration of minimal energy for each fixed number of positive spins. We compare the results from the algorithm to the results from the direct analysis between the proposed shapes.

Thirdly, we want to study the first hitting time of the stable state under the Metropolis-Glauber dynamics. We want to use the first hitting time to estimate the energy barrier between the metastable state and the stable state. We look at for which values of β we see metastable behaviour.

1.4 Organization

In chapter 2 we formally define our setting and compare the setting to the regular Ising Model in 2D and the long range Ising Model in 1D. In chapter 3 we compare the energy of two possible candidates for the critical droplet, the circle droplet and the square droplet. In chapter 4 we use this information to try to find a formula for the size of the critical radius. In chapter 5 we look at the results of simulations of paths from $-\mathbb{1}$ to $+\mathbb{1}$ to look at the critical droplet for each of these simulations, and compare them to our results on the square and the circle. In chapter 6 we simulate the Metropolis-Glauber dynamics on our lattice and look at the results of hitting times and their scaling. In chapter 7 we discuss the results, difficulties and possibilities for future research.

2 Ising Model in 2D

We start with some definitions, to allow us to talk more clearly about the topic of the thesis, its goals and its setting.

2.1 Definitions

Our setting is based on [8] about nucleation for one-dimensional long-range Ising models.

We have a large square lattice $\Lambda_L \subset \mathbb{Z}^2$ of size $L \times L$. We define the space of states on Λ_L as $\Omega_L = \{-1, 1\}^{\Lambda_L}$. Our distance is the Euclidean distance with periodic boundary conditions. Formally, our distance is given by

$$d(x,y) = \min_{i \in \{-1,0,1\}} \min_{j \in \{-1,0,+1\}} d'(x + (i * L, j * L), y)$$

where $d'(x,y)$ is the regular Euclidean distance in $\mathbb{R}^2$.

For a configuration σ of spins on $\Lambda \coloneqq L \times L$, our Hamiltonian is given by

$$H_{\Lambda,h}(\sigma) = -\sum_{\{i,j\} \subset \Lambda} J(i,j)\sigma_i\sigma_j - \sum_{i \in \Lambda} h\sigma_i \tag{4}$$

We choose as $J(i,j) = Jd(i,j)^{-\lambda}$, where our distance is the distance as defined above.

In this case, we see that $+\mathbb{1}$ is the state which minimizes the Hamiltonian, since it minimizes both terms of the formula for the Hamiltonian (the interaction term and the external magnetic field term).

In order to formulate our goals, we need to introduce dynamics on our system.

2.1.1 Markov Chain

Our dynamics are defined via a discrete time Markov Chain. Remember that our space of states is $\Omega_L = \{+1, -1\}^{\Lambda_L}$, where Λ_L is a lattice of size $L \times L$.

We first want to define the Markov define heuristically. At every time-step, we select a random vertice to flip its spin. If this spin flip would lower the Hamiltonian, the spin on the vertice is always flipped. If the spin flip would increase the Hamiltonian, the spin is flipped with probability $e^{-\beta \Delta H}$. ΔH is the difference in the Hamiltonian that flipping the spin makes, and so is always negative in this case, and β is a variable which in statistical physics is inversely related to the temperature.

We want to define this formally.

Let $\eta, \eta' \in \Omega_L$. First, let $d(\eta, \eta') = \#i \in \Lambda_L$ such that $\eta_i \neq \eta'_i = 1$. We define a function $q : \Omega_L \times \Omega_L \to \mathbb{R}_+$.

$$q(\eta, \eta') = \begin{cases} \frac{1}{L^2} & \text{if } d(\eta, \eta') = 1 \\ 0 & \text{otherwise} \end{cases}$$

Now, we can define $P(\eta, \eta')$ as

$$P(\eta, \eta') = \begin{cases} q(\eta, \eta') \min[1, e^{-\beta(H(\eta') - H(\eta))}] & \text{if } \eta \neq \eta' \\ 1 - \sum_{\zeta \in \Omega} q(\eta, \zeta) \min[1, e^{-\beta(H(\zeta) - H(\eta))}] & \text{if } \eta = \eta' \end{cases}$$

We note that lower β increases the chaotic behaviour in the system, since for lower β, the probability that a spin is flipped is higher than it would be for hiogher β. We note that at each time step, we flip only a single spin. Therefore, if two states differ on the spin value on k vertices, it would take at least k time steps for the system to transition from one state to the other.

This Markov chain is reversible with respect to the Gibbs measure,

$$\mu_{\Lambda_L}(\eta) = \frac{1}{Z_{\Lambda_L}} e^{-\beta H_{\Lambda_L,h}(\sigma)},$$

where Z_{Λ_L} is a normalizing constant.

A Markov chain on a finite space of states is reversible with respect to a measure if it satisfies the balance equation

$$\mu(\eta)P(\eta,\eta') = \mu(\eta')P(\eta',\eta)$$

[1], for all $\eta \in \Omega_L$. A measure with respect to which the Markov chain is reversible is also the stationary measure of that Markov chain [6]. Therefore, the Gibbs measure is the stationary measure for our Metropolis-Glauber dynamics.

The state of the Markov chain so defined at time t and started in η is denoted by σ_t^η.

2.1.2 Metastability

We see that on our Markov chain, the probability of going to an adjacent state with a higher Hamiltonian is proportional to $e^{-\beta\Delta H}$. Therefore, as in [5], to look at metastable regions we look at the height of the energy barrier between a state and the stable set.

The stable set is the set with the lowest Hamiltonian. Formally, we define the set of stable sets as

$$\mathscr{X}^s = \arg\min_{\{\eta\in\Omega_L\}} H(\eta).$$

In our case, $\mathscr{X}^s = \{_{\mathbb{1}}\}$, since $+\mathbb{1}$ is the only term to minimize both the interaction term and the external magnetic field term of the Hamiltonian.

We define a path $\gamma = \{\eta_0, \ldots, \eta_k\}$ as a sequence of states such that two consecutive states η_j, η_{j+1} only differ on the value of one spin. A path $\gamma, \sigma \to \eta$ is a path $\gamma = \{\eta_0, \ldots, \eta_k\}$ such that $\gamma_0 = \sigma, \gamma_k = \eta$. We can now define the communication height. The communication height between σ and η is the lowest possible maximum Hamiltonian over all paths from a state σ to η.

$$\Phi(\sigma,\eta) = \min_{\gamma:\sigma\to\eta}\max_{\psi\in\gamma} H_{\Lambda,h}(\psi),$$

where we take the minimum over all paths from $\sigma \to \eta$.

For a state η, we define $\mathscr{I}_\eta = \{\sigma \in \Omega_L : H(\sigma) < H(\eta)\}$. If $\eta \notin \mathscr{X}^s$, we define the stability level of η as $V_\eta = \Phi(\eta, \mathscr{I}_\eta) - H(\eta)$.

We can therefore formally define the set of metastable states as

$$\mathscr{X}^m := \{\sigma \in \Omega_L : V_\sigma = \max_{\eta\in\Omega_L} V_\eta\}.$$

We can visualize the metastable state as the second deepest well of the Hamiltonian, as in figure 1. The metastable state can be understood as the most stable state outside of the stable state, with its own basin of attraction.

In our system, we expect $-\mathbb{1}$ to be the metastable state. In this thesis, we do not prove that $-\mathbb{1}$ is the actual metastable state. We expect $-\mathbb{1}$ to be the metastable state since it has a deep minimum in its Hamiltonian, since it minimizes the first term but maximizes the second term. To transition to the stable state, $+\mathbb{1}$, it has to travel over an energy bump in which neither term is maximized. Therefore, we expect $-\mathbb{1}$ to have a deep minimum, and we expect it to be the most stable state apart from the actual stable state.

Conjecture 1. $\mathscr{X}^m = \{-\mathbb{1}\}$.

We are interested in the path between the metastable state and the stable state. Therefore, we have to define the energy barrier between the metastable state and the stable state.

$$\Gamma = \Phi(\mathscr{X}^m, \mathscr{X}^s) - H(\mathscr{X}^s). \tag{5}$$

If we assume the conjecture is true, we get $\Gamma = \Phi(-\mathbb{1}, +\mathbb{1}) - H(-\mathbb{1})$ for our model.

We will also look at the critical droplets. Critical droplets are the states with energy level $\Gamma + H(-\mathbb{1})$ which are visited before the stable state with probability 1 as β tends to ∞. They are a set of states which exist on the edge of the basin of attraction for both the metastable state and the stable state. As the

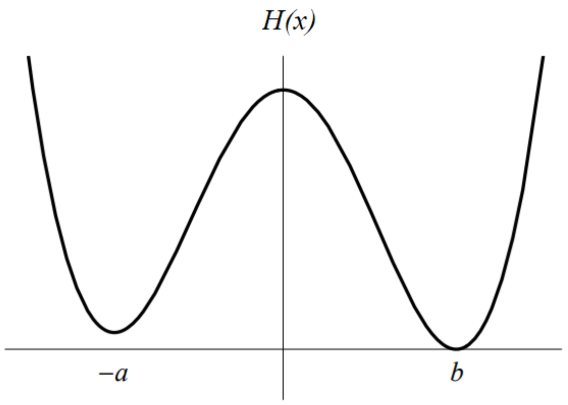


Figure 1: A sketch of a double well with two local minima and one global minimum. [7]

temperature goes to 0, the critical droplets are always visited on the most efficient paths. We can visualize the critical droplet as the state with the maximum Hamiltonian between the two wells in figure 1.

For a specific path γ, we define

$$\Phi_\gamma = \max_{\psi \in \gamma} H_{\Lambda,h}(\psi)$$

We use this in our definition of critical configurations. We can formally define the critical configurations as the set of configurations $\mathscr{P}_{\text{critical}}$ such that

- $H(\eta) - H(-\mathbb{1}) = \Gamma$ for all $\eta \in \mathscr{P}_{\text{critical}}$.
- Any path γ from $-\mathbb{1}$ to $+\mathbb{1}$ such that $\Phi_\gamma = \Gamma$ visits $\mathscr{P}_{\text{critical}}$.
- $\lim_{\beta\to\infty} \mathbb{P}(\tau^{-\mathbb{1}}_{\mathscr{P}_{\text{critical}}} < \tau^{-\mathbb{1}}_{+\mathbb{1}}) = 1$.

Since the difference of the Hamiltonian between a state and the metastable state is often important, we define

$$E(\sigma) = H(\sigma) - H(-\mathbb{1}) \tag{6}$$

Therefore, for any $\sigma_{\text{critical}} \in \mathscr{P}_{\text{critical}}, \Gamma = E(\sigma_{\text{critical}})$.

The hitting time τ^η_ζ is formally defined as $\min\{t > 0 : \sigma^\eta_t = \zeta\}$, where σ^η_t was the Markov chain starting in η, as previously defined. In the thesis, we will study the hitting time of $+\mathbb{1}$ when starting in the state $-\mathbb{1}$. We want to look at the relationship between this energy barrier and the hitting time of the stable state. Theorem 4.9 in [5] states

Theorem 1.. For any metastable state η,

$$\lim_{\beta\to\infty} \frac{1}{\beta} \log \mathbb{E}\tau^\eta_{\mathscr{X}^s} = \Gamma \tag{7}$$

Under assumption of conjecture 2.1.2, this gives us $\lim_{\beta\to\infty} \frac{1}{\beta}\log(\mathbb{E}[\tau^{-\mathbb{1}}_{+\mathbb{1}}]) = \Gamma$ by filling in the metastable state and the stable state.

We can use this theorem to study the energy level of the critical droplets if we have a good estimate of the expected hitting time for asymptotic β.

Certain sets of states are important in the thesis. We want to look at the Hamiltonian of states which consist of a square of positive spins of size $l \times l$ at the center of the larger square, with the rest of spins being negative. We denote this state by $\sigma_l \in \Omega_{\text{sq}}$.

Similarly, we want to look at the set of states where all spins inside a circle of radius r have positive spin, with the rest of the spins being negative. We denote these states by $\eta_r \in \Omega_{\text{circ}}$.

Finally, we want to look at the set of states where all spins inside a λ-circle of radius r have positive spin, with the rest of the spins being negative.

Recall that the λ-norm is defined as

$$||x-y||_\lambda = (|x_1-y_1|^\lambda + |x_2-y_2|^\lambda)^{\frac{1}{\lambda}}$$

We denote these states by $\eta_r \in \Omega_{\lambda\text{-circ}}$.

We study these states since they are close to the possible configurations for the critical droplets. They are connected, convex and they have the same width as length. These are all properties which we would expect the critical droplet to have.

2.2 Comparison to short-range 2D Ising Model

In 2D, we know that the critical droplet is a semi-rectangle of size $\lceil 2 * J/h \rceil$ with a protuberance added to one of the sides, by theorem 4.29 in [4]. Since in our thesis we set J to one, this simplifies to $\frac{2}{h}$.

We want to compare the critical droplet in our setting to the critical droplet in this model. Moreover, we expect the critical droplet for $\lambda \to \infty$ to tend towards this formula, since for high λ, the long-range Ising model tends towards the short-range Ising model.

Concerning the values we choose for our parameters, there are some differences between the short-range Ising Model in 2D and the long-range Ising Model in 2D. For example, we can look at the spin flip, where we change the value of the spin on one of the vertices and compare the Hamiltonian between the state before the spin was flipped and after the spin was flipped. We can look at the maximum difference flipping a spin can make for both of the terms in the Hamiltonian. For the second term, this change is the same for both the short-range model and the long-range model at $2h$. Therefore, the first term is the most interesting term. For the short-range Ising model, $|\Delta H|$ for one spin flip can be bound by $|\Delta H| \leq 8J + 2h$. Since one spin can only interact with its four neighbours the maximum difference in the interaction term is $8J$, and the flipping of one spin causes a maximum difference of $2h$ in the term for the external magnetic field. For the long-range system, the value of the spin flip is not as easy to bound. If we look at our system of a lattice of size 101×101, the spin flip could be of size

$$4 * \sum_{a=1}^{50} \sum_{b=0}^{50} \frac{2J}{(a^2+b^2)^{\lambda/2}} + 2h.$$

If we choose $J = 1, \lambda = 2$, this spin flip has $\Delta H \approx 56 + 2h$, whereas the spin flip in the short-range Ising model would be bound by $8 + 2h$. This is a rather large difference and we have to be careful with selecting the values of our parameters, since the suitable regions could differ from the short-range Ising model. In the short-range Ising model, if we choose $J = 1, h = 2$, our critical droplet would be the size of a single vertice. For the long-range Ising model, this might not be the case, and therefore we have to investigate a larger range of possibilities.

2.3 Comparison to long-range 1D Ising Model

There has been research on the Ising Model in 1D [8]. This paper looks at the difference between the long-range Ising Model and the short-range Ising model in 1D.

To recall, the formula for the interaction for the short-range Ising model is

$$J(||i-j||) = \begin{cases} 1 \text{ if } ||i-j|| = 1 \\ 0 \text{ otherwise.} \end{cases}$$

We see that the interaction term is dependent only on the border between the positive component and the negative component. In 1D, every connected positive component has a constant size of its border, namely 2. Because of this, there is no real critical interval in the short-range Ising model in 1D; as soon as one spin is positive, the system immediately nucleates the stable phase.

For the $1D$ model with long-range interaction, this is not the case. [8] finds that the $1D$ long-range Ising model does have a critical droplet, of size

$$k_c \sim \left(\frac{J}{h(\lambda - 1)} \right)^{\frac{1}{\lambda-1}}$$

We use this formula to try to deduce a formula for the critical diameter in the long-range $2D$ model. The formulas we try to fit are equations 8, 10 and 11. Formula 10 is a best guess based on the formula for the $1D$ model.

2.4 Design of the model and choice of parameters

The larger we choose l, the better for the asymptotic behaviour of the model. We try to study situations where $L \gg d$, where d is the diameter of our positive component.

If this is not the case, the size of the lattice starts to interfere with the size of the positive component. For any formula for the size of the critical droplet, the formula cannot be correct if it predicts a critical droplet of a size greater than the size of the lattice. Therefore, for relevant critical droplets we need $d \ll L$.

On the other side, we have the constraint that our calculations comparing the circle and the square are not useful if the critical droplet is too small. For small critical droplets, the size of the border starts to dominate the interaction term of the Hamiltonian and the optimum shape will tend to the shape which minimizes this border, as in the normal $2D$ short-range Ising model. For relevant critical droplets, we need $1 \ll d$.

Therefore, we want to look at states for which the positive component is large enough to not be influenced too much by microscopic behaviour, but is not large enough to be influenced by the macroscopic behaviour of the lattice size. We look at a mesoscopic scale which is an order of magnitude larger than the size of an individual spin, but about an order of magnitude smaller than the lattice size.

We therefore choose $L = 101$. For this size we expect to be able to find mesoscopic critical droplets, since we expect there to be a decent range of diameters d such that $1 \ll d \ll L$. This would allow us to study the model for a large range of parameter values.

3 Disc vs Quasi Rectangle

We expect that a main difference between the short-range Ising Model and the long-range Ising model is the shape of the critical droplet. For the long-range Ising model, we expect the critical shape to be close to a circle, for appropriate choices of parameters. For the optimal path from $-\mathbb{1}$ to $+\mathbb{1}$, we have to minimize the border between the positive component and the negative component for each number of positive spins. In the short-range Ising model, this border is simple, only neighbouring vertices interact. For the long-range Ising model, the border is more complicated to analyze, and we have to deal with a long-range isoperimetric problem. Since the vertices interact over longer distances, minimizing their distance towards each other and maximizing their distance towards all other vertices is important. Therefore, we suspect a circle is a more efficient shape.

The circle on the lattice is a set of vertices η_r on the lattice such that for some origin $i \in \Lambda$, η_r contains all vertices within distance r of i and no vertices at a larger distance than r from i.

In C++ we calculate the Hamiltonian for a variety of states to analyze its behaviour and differences between states with differently shaped positive components.

We construct an array of size $L \times L$, where we start by choosing $L = 101$. We fill this array with values 1 and -1 to represent a grid of spins. For this purpose we construct functions to fill the array in this matter, with the positive components in several shapes, such as the square, the circle and the λ-circle. For a given radius and lattice size, these functions take an origin vertex on the lattice and for all vertices within the selected radius of the relevant distance of this origin vertex it sets the spins to $+1$. For all other vertices, the spins are set to -1. For each shape there is a distance function for which this construction results in positive components of the correct shape; the Manhattan distance for the square, the Euclidean distance for the circle and the p-norm with $p = \lambda$ for the λ-circle.

Thereafter we construct a function which evaluates the Hamiltonian on these states with periodic boundary conditions for different values of J, h and λ. This function has the chosen parameters and the previously generated lattice as input. It calculates the Hamiltonian by starting the Hamiltonian at 0, summing over all combinations of vertices to add the appropriate interaction term to the total, and for each individual vertex adding the appropriate value for the external magnetic field term to the total. We evaluate the Hamiltonian for different sizes of the positive components (different radii for the square and circle). We export the values to a .csv file, which we evaluate in R.

The code for calculating the Hamiltonian can be found in appendix A.

3.1 Results

We see that the Hamiltonian does indeed follow the shape we expect. We have calculated the Hamiltonian for all radii using the code in appendix A, described in section 3, for the values in tables 1 through 6, but we do not show the curve of the Hamiltonian as dependent on the radius for all combinations of values.

For a typical example of the function of the Hamiltonian with respect to the different shapes for different radii, we look at the graph for $J = 1, h = 0.6$ and $\lambda = 2.5$ in figure 2.

We notice the increase in Hamiltonian compared to the value of $H(-\mathbb{1})$ and then the lower Hamiltonian at the end, for $H(+\mathbb{1})$. For this initial example, we see that the circle indeed has a lower maximum Hamiltonian than the square, although the λ-circle is very close to it. We also note the behaviour of the size for the square, where it suddenly flattens for radii above 50. This is because the square with radius 50 fills the entire lattice. This is not true for the circle and λ-circle, which is why there is a difference in behaviour at the higher radii.

In section 3.1.2 we discuss the most efficient shapes for different combinations of h and λ.

3.1.1 Critical Droplet size

With respect to the size of the critical droplet, the results of our calculations are in tables 1 through 6 as well. We have to note some specifics about the results in the table. For each combination of h and λ we have two values. The first value is the critical radius. The critical radius is the radius for which the shape had the highest Hamiltonian for the chosen combination of parameters, the 'least efficient' radius for the corresponding shape. We recall that for a given radius, the component consists of all elements which are within the relevant distance to some chosen origin point. Because of periodic boundary conditions, the choice of this origin is irrelevant for the Hamiltonian. It is important to note that a component with shape 0 still

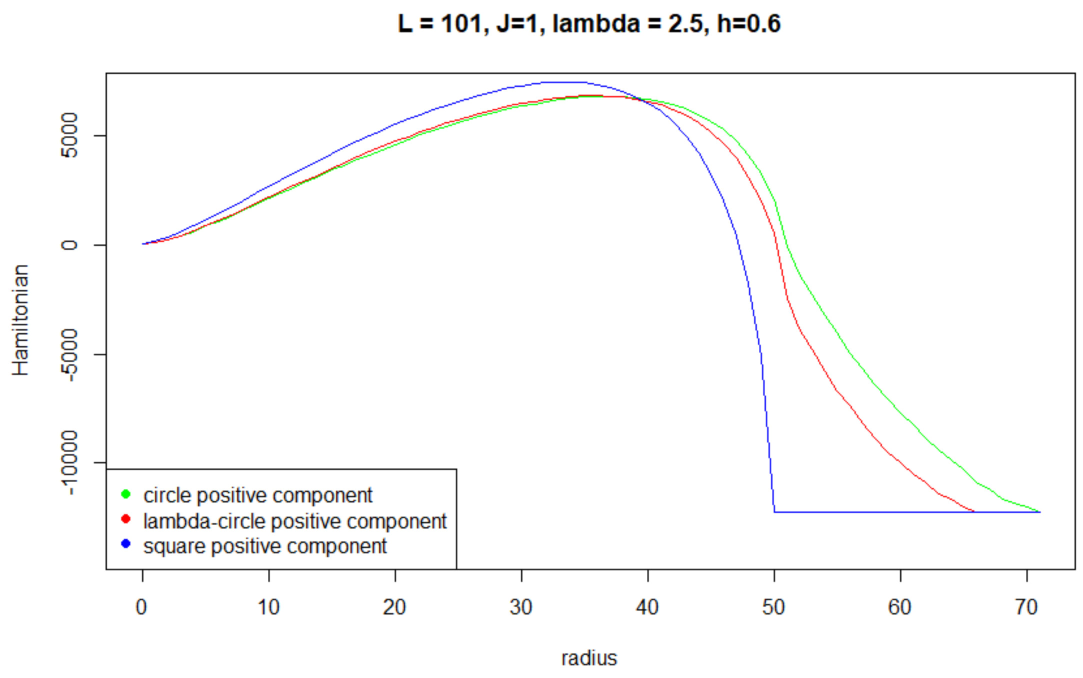


Figure 2: The Hamiltonian for different shapes for $J = 1, h = 0.6$ and $\lambda = 2.5$

contains the origin element, and so the state consisting of a component with radius 0 is NOT $-\mathbb{1}$. If $-\mathbb{1}$ is the state with the highest Hamiltonian, we have denoted this with radius -1.

If the radius is $x.50$, the Hamiltonian was equal for x and $x + 1$.

The second value is the difference between the Hamiltonian of this state and $-\mathbb{1}$.

So, if our set of states is $\Omega_{\text{sq, circ, or } \lambda\text{-circ}}$, the first value is $\arg\max_r[H(\eta_r)]$. Hence, we have $H(\eta_{r'}) \in \Omega$, $\forall r'$.

For this selected η_r, the second value would then be $E(\eta_r) = H(\eta_r) - H(-\mathbb{1})$.

3.1.2 Most efficient shape for different values

For the values of λ, h we have examined, we want to look at which shape has the lowest maximum Hamiltonian, since it is very likely that the optimal path from $-\mathbb{1}$ to $+\mathbb{1}$ would use these shapes.

We will first compare the circle and the square and afterwards look at the λ-circle and its impact.

We see that as λ increases, the range for h for which the circle is the most efficient shape decreases compared to the square.

For $\lambda = 2.0$, the circle is the most efficient shape even for $h = 15.0$. This steadily decreases as λ increases. We have summarized this in table 7.

This indicates that there is not one range for λ for which the circle is more efficient than the square, but that this is a function of h as well.

We can view this even more clearly in figure 3 and figure 4.

These figures show us which shape has the lowest maximum Hamiltonian for values of h and λ. Figure 3 compares the circle and the square. We see a top-right triangle for low values of h and or λ for which the circle is more efficient than the square. This is at least what we expect for smaller λ, since as λ tends to infinity we expect our model to resemble the standard $2D$ Ising Model.

To explain the dependency on h, we can look at the size of the critical droplet for h and λ. We see that as h increases, the radius of the least efficient droplets decreases. The relationship is more complicated when looking at λ, since the diameter increases for increasing λ for some values of (h, λ), but decreases for other values of the parameters. An important factor here seems to be whether we are in the macroscopic or

	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3.0	3.1	3.2
0.01	43.00 70269.00	43.00 51875.00	43.00 38603.00	44.00 29000.90	44.00 22026.20	44.00 16909.70	45 13137.6	45.00 10354.10	46.00 8272.30	46.00 6715.40	46.00 5529.50	47.00 4627.70	47.00 3930.30	47.00 3384.00
0.02	43.00 70153.00	43.00 51759.20	43.00 38487.20	44.00 28879.40	44.00 21904.70	44.00 16788.20	44 13012.8	45.00 10226.90	45.00 8142.20	46.00 6582.90	46.00 5396.90	47.00 4489.20	47.00 3791.90	47.00 3245.60
0.04	43.00 69922.00	43.00 51527.70	43.00 38255.60	44.00 28636.30	44.00 21661.50	44.00 16545.10	44 12769.7	45.00 9972.50	45.00 7887.80	46.00 6317.90	46.00 5131.90	46.00 4221.00	46.00 3515.50	47.00 2968.70
0.08	43.00 69459.00	43.00 51063.60	43.00 37792.60	43.00 28171.10	44.00 21175.40	44.00 16058.90	44.00 12283.60	44.00 9479.90	44.00 7383.80	45.00 5806.30	45.00 4610.70	45.00 3694.20	45.00 2985.70	46.00 2434.10
0.15	42.00 68666.00	43.00 50253.20	43.00 36982.20	43.00 27360.70	43.00 20351.70	43.00 15217.00	43.00 11433.50	44.00 8629.20	43.00 6533.50	43.00 4958.50	43.00 3765.00	42.00 2858.50	41.00 2164.30	39.00 1646.80
0.30	42.00 67008.00	42.00 48565.10	42.00 35270.10	42.00 25642.60	42.00 18635.40	42.00 13507.80	42.00 9733.90	41.00 6947.90	39.00 4899.50	37.50 3398.40	34.00 2333.60	30.00 1600.80	25.00 1120.60	22.00 814.80
0.60	42.00 63693	42.00 45250.40	41.00 31960.1	41.00 22374.60	40.00 15432.30	39.00 10409.50	37.00 6805.6	34.00 4302.80	29.00 2659.40	23.00 1650.80	19.00 1054.1	16.00 702.10	13.00 491.4	11.00 356.30
1.2	40 57292	40.00 39001.20	3, 3, 25903.3	37.00 16589.30	34.00 10120.20	30.00 5838.40	24 3242.8	19.00 1799.90	15.00 1026.90	12.00 614.40	9 391.2	8.00 264.20	6 188.9	5.00 142.60
2	39 49382	38.00 31417.60	35 18967.8	31.00 10661.50	25.00 5596.50	20.00 2826.00	15 1427.7	11.00 750.50	8.00 423.70	6.00 254.80	5 167	4.00 116.50	3 85.8	3.00 67.20
3	37 40367	34.00 23425.60	29 12454.1	23.00 6091.60	17.00 2834.20	12.00 1298.60	9 622.6	6.00 322.90	5.00 182.50	4.00 112.90	3 77.9	2.00 53.90	2 42.3	2.00 32.80
5	31 25862.7	25.00 12388.30	19 5341.6	13.00 2150.50	9.00 853.10	6.00 360.50	4 170.5	3.00 90.60	2.00 53.34	2.00 32.58	1 22.38	1.00 16.56	1 11.69	1.00 7.60
7	25 16013.9	18.00 6479.70	12 2348.2	8.00 824.80	5.00 303.20	3.00 126.01	2 59.22	1.00 28.45	1.00 17.90	1.00 9.40	0 5.1	0.00 3.90	0 2.8	0.00 2.00
10	18 7590.7	11.00 2428.90	7 715	4.00 177.30	2.00 73.60	1.00 27.80	1 11.5	0.00 4.40	0.00 2.20	0.00 0.50	-1 0	-1.00 0.00	-1.00 0.00	-1.00 0.00
15	9 2031.4	5.00 461.70	2 96.1	1.00 24.90	0.00 4.70	0.00 0.50	-1 0	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00

Table 1: The size of the least efficient circle and the corresponding energy for $\lambda = 1.9$ to $\lambda = 3.2$

	3.3	3.4	3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	5.0	6.0	7.0	100.0
0.01	48.00	48.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00	50.00	50.00	50.00	50.00
	2958.30	2619.00	2347.70	2129.80	1950.00	1800.10	1674.30	1567.80	1476.90	1398.60	1036.90	858.40	773.90	643.10
0.02	48.00	48.00	48.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00	49.00
	2814.00	2474.70	2200.70	1979.30	1799.40	1649.70	1523.80	1417.30	1326.40	1248.10	885.70	704.80	620.40	491.00
0.04	47.00	47.00	47.00	47.00	48.00	48.00	48.00	48.00	48/49	49.00	46.00	42.00	38.00	31.00
	2536.90	2192.20	1914.60	1688.90	1505.10	1352.60	1224.90	1117.10	1025.30	947.10	587.30	421.20	354.70	264.00
0.08	46.00	46.00	45.00	44.00	43.00	39.00	37.00	37.00	34.00	32.00	24.00	21.00	18.00	16.00
	1999.20	1653.00	1375.70	1158.40	979.30	840.40	728.70	639.50	572.40	515.00	297.90	215.30	182.00	136.50
0.15	36.00	32.00	29.00	27.00	23.00	22.00	21.00	20.00	17.00	17.00	13.00	12.00	10/11	8.00
	1257.00	980.10	781.70	635.00	528.80	452.30	389.90	342.20	308.60	279.70	162.90	118.90	99.80	76.90
0.30	18.00	17.00	15.00	13.00	12.00	11.00	10.00	10.00	8.00	8.00	7.00	6.00	5.00	4.00
	607.80	474.60	375.20	309.30	259.40	221.90	193.50	171.10	153.90	141.30	85.40	62.40	53.20	42.60
0.60	9.00	8.00	8.00	7.00	6.00	5.00	5.00	5.00	4.00	4.00	3.00	3.00	3.00	2.00
	269.60	214.80	173.60	144.50	123.20	106.80	95.10	85	76.5	71	45.4	34.6	29.4	24.4
1.2	4.00	4.00	4.00	3.00	3.00	3.00	3.00	2	2	2	2	2	1	1
	109.70	90.50	74.60	64.90	56.90	50.00	44.10	39.5	36.9	34.7	23.1	16.7	14.7	12
2	3.00	2.00	2.00	2.00	2.00	2.00	2.00	1	1	1	1	1	1	0/1
	51.70	43.80	38.10	33.00	28.70	24.90	21.60	19.7	18.56	17.48	11.82	8.47	6.75	4
3	2.00	1.00	1.00	1.00	1.00	1.00	1.00	1	1	1	0	0	0	0
	24.68	21.11	18.54	16.29	14.35	12.62	11.11	9.758	8.555	7.48	4.18	3.32	2.84	2
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	0	0	0	-1	-1	-1
	5.20	4.50	4.00	3.50	3.10	2.70	2.40	2	1.8	1.6	0.2	0	0	0
7	0.00	0.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	1.20	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 2: The size of the least efficient circle and the corresponding energy for $\lambda = 3.3$ to $\lambda = 100.0$

	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3.0	3.1	3.2
0.01	43.00 70182.00	43.00 51875.00	43.00 38679.80	43.00 29099.20	43.00 22137.90	43.00 17050.70	43.00 13263.80	44.00 10495.30	44.00 8414.00	45.00 6837.40	45.00 5627.30	45.00 4735.40	46.00 4030.90	46.00 3466.50
0.02	43.00 70069.00	43.00 51759.20	43.00 38561.60	43.00 28979.30	43.00 22016.20	43.00 16927.30	43.00 13139.30	44.00 10363.30	44.00 8280.90	45.00 6696.90	45.00 5485.90	45.00 4592.80	46.00 3881.10	46.00 3316.10
0.04	43.00 69842.00	43.00 51527.70	43.00 38325.30	43.00 28739.40	43.00 21772.80	43.00 16680.40	43.00 12890.30	44.00 10099.50	44.00 8014.60	44.00 6428.00	44.00 5207.50	45.00 4307.60	45.00 3588.90	45.00 3016.80
0.08	43.00 69388.00	43.00 51063.60	43.00 37852.60	43.00 28259.70	43.00 21286.00	43.00 16186.50	43.00 12392.30	43.00 9588.40	43.00 7489.50	43.00 5900.10	43.00 4669.40	43.00 3761.30	44.00 3037.30	43.00 2456.80
0.15	43.00 68593.00	43.00 50253.20	42.00 37041.50	42.00 274433.40	42.00 20467.50	42.00 15327.60	42.00 11521.00	42.00 8725.50	42.00 6609.30	42.00 5021.70	41.00 3794.70	41.00 2891.20	39.00 2189.80	37.00 1637.90
0.30	42.00 66892.00	42.00 48565.10	42.00 35348.40	42.00 25728.60	42.00 18728.40	41.00 13593.50	41.00 9791.00	40.00 7015.10	38.00 4941.20	36.00 3429.30	32.00 2327.70	28.00 1611.90	24.00 1130.40	20.00 800.40
0.60	42.00 63635.00	42.00 45250.10	41.00 32022.00	40.00 22427.50	39.00 15487.20	38.00 10448.50	36.00 6820.10	32.00 4327.40	28.00 2672.90	22.00 1663.60	18.00 1064.20	14.00 708.00	12.00 495.50	10.00 346.10
1.2	41.00 57253.00	40.00 39001.20	39.00 25924.60	37.00 16605.30	34.00 10142.70	29.00 5850.90	24.00 3239.80	19.00 1812.40	14.00 1032.90	11.00 623.00	9.00 397.00	7.00 270.80	6.00 191.50	4.00 137.50
2	39.00 49332.00	38.00 31417.60	35.00 18982.10	31.00 10652.60	24.00 5596.70	19.00 2838.70	14.00 1422.30	10.00 755.70	8.00 425.70	6.00 259.10	5.00 167.00	4.00 119.20	3.00 85.80	3.00 67.20
3	37.00 40355.00	34.00 23425.60	29.00 12449.40	23.00 6075.30	17.00 2865.50	12.00 1301.60	8.00 611.10	6.00 328.70	4.00 184.40	4.00 114.20	3.00 77.90	2.00 53.90	2.00 42.30	2.00 32.80
5	31.00 25867.10	25.00 12388.30	19.00 5345.50	13.00 2128.90	9.00 854.80	6.00 360.50	4.00 174.10	3.00 90.60	2.00 53.34	2.00 32.58	1.00 22.38	1.00 16.56	1.00 11.69	1.00 7.60
7	25.00 16012.00	18.00 6479.70	12.00 2360.20	8.00 821.40	5.00 303.20	3.00 126.01	2.00 59.22	1.00 28.45	1.00 17.90	1.00 9.40	0.00 5.10	0.00 3.90	0.00 2.80	0.00 2.00
10	18.00 7595.20	11.00 2428.90	7.00 718.30	4.00 177.30	2.00 73.60	1.00 27.80	1.00 11.50	0.00 4.40	0.00 2.20	0.00 0.50	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00
15	10.00 2033.00	5.00 461.70	2.00 96.10	1.00 24.90	0.00 4.70	0.00 0.50	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00	-1.00 0.00

Table 3: The size of the least efficient λ-circle and the corresponding energy for $\lambda = 1.9$ to $\lambda = 3.2$

	3.3	3.4	3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	5.0	6.0	7.0	100.0
0.01	47.00	47.00	47.00	48.00	48.00	48.00	49.00	49.00	49.00	49.00	49/50	50.00	50.00	50.00
	3052.00	2705.10	2430.8	2188.40	2018.40	1863.00	1715.90	1624.70	1531.80	1454.20	1054.60	868.40	769.80	596.00
0.02	46.00	46.00	47.00	47.00	48.00	48.00	48.00	48.00	48.00	48.00	49.00	49.00	49.00	50.00
	2894.50	2548.00	2271.20	2022.60	1850.60	1694.50	1549.30	1453.90	1360.30	1276.70	873.80	684.00	583.00	400.00
0.04	46.00	46.00	46.00	46.00	46.00	47.00	46.00	47.00	47.00	46.00	42.00	35/36	31.00	24.00
	2591.20	2243.50	1955.20	1703.00	1529.90	1369.70	1218.90	1122.70	1023.70	940.20	533.90	381.40	313.20	204.80
0.08	43.00	43.00	43.00	41.00	40.00	38.00	36.00	34.00	31.00	29.00	21.00	18.00	17.00	12.00
	2026.00	1671.70	1383.60	1138.90	974.50	829.70	700.10	627.90	557.60	501.40	265.50	198.50	163.20	104.50
0.15	34.00	31.00	27.00	25.00	23.00	20.00	20.00	18.00	17.00	16.00	11.00	9.00	9.00	7.00
	1265.70	979.60	779.30	617.40	527.30	447.60	368.80	339.20	300.90	272.70	141.80	112.50	93.90	58.50
0.30	18.00	16.00	13.00	13.00	11.00	11.00	10.00	9.00	9.00	8.00	4.00	5.00	5.00	4.00
	612.90	474.70	376.70	293.20	260.90	222.20	177.90	173.50	154.70	140.30	74.90	62.10	52.40	30.60
0.60	9.00	7.00	7.00	6.00	5.00	5.00	5.00	5.00	4.00	4.00	3.00	3.00	3.00	1.00
	273.50	213.70	175.80	130.80	124.30	109.90	84.30	84.00	78.00	72.10	45.40	34.60	29.40	18.00
1.2	4.00	4.00	4.00	3.00	3.00	3.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00	1.00
	112.80	92.30	75.30	64.90	56.90	50.00	42.40	39.50	36.90	34.70	23.10	16.70	14.70	12.00
2	3.00	2.00	2.00	2.00	2.00	2.00	2.00	1.00	1.00	1.00	1.00	1.00	1.00	0/1
	51.70	43.80	38.10	33.00	28.70	24.90	21.60	19.70	18.56	17.48	11.82	8.47	6.75	4.00
3	2.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
	24.68	21.11	18.54	16.29	14.35	12.62	11.11	9.76	8.56	7.48	4.18	3.32	2.84	2.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00	-1.00	-1.00
	5.20	4.50	4.00	3.50	3.10	2.70	2.40	2.00	1.80	1.60	0.20	0.00	0.00	0.00
7	0.00	0.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	1.20	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 4: The size of the least efficient λ-circle and the corresponding energy for $\lambda = 3.3$ to $\lambda = 100.0$

	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3.0	3.1	3.2
0.01	38.00	38.00	39.00	39.00	40.00	40.00	40.00	41.00	41.00	42.00	43.00	43.00	44.00	44.00
	74303.20	55042.80	41167.60	31061.80	23670.70	18260.40	14244.30	11271.60	9033.30	7353.40	6067.00	5087.80	4324.00	3730.90
0.02	38.00	38.00	39.00	39.00	40.00	40.00	40.00	41.00	41.00	42.00	42.00	43.00	43.00	44.00
	74184.60	54924.20	41042.80	30937.00	23539.50	18129.20	14113.10	11133.80	8895.60	7208.90	5922.00	4936.30	4169.90	3572.50
0.04	38.00	38.00	39.00	39.00	39.00	40.00	40.00	41.00	41.00	41.00	42.00	42.00	43.00	43.00
	73947.50	54687.10	40793.20	30687.40	23286.70	17866.80	13850.70	10858.30	8620.00	6920.40	5633.00	4637.40	3867.20	3262.20
0.08	38.00	38.00	39.00	39.00	39.00	40.00	40.00	40.00	40.00	41.00	41.00	41.00	41.00	41.00
	73473.20	54211.80	40294.00	30188.20	22757.50	17341.80	13325.90	10325.70	8069.50	6369.20	5068.70	4065.60	3285.40	2673.60
0.15	38.00	38.00	38.00	39.00	39.00	39.00	39.00	39.00	39.00	39.00	39.00	38.00	37.00	36.00
	72642.20	53381.80	39435.40	29314.50	21913.70	16463.20	12425.50	9415.50	7156.30	5448.60	4147.80	3150.90	2388.00	1806.50
0.30	38.00	38.00	38.00	38.00	38.00	38.00	38.00	37.00	36.00	34.00	32.00	28.00	24.00	20.00
	70863.60	51603.30	37655.90	27509.30	20089.60	14634.60	10600.60	7612.50	5392.60	3754.40	2574.20	1758.40	1220.50	872.70
0.60	37.00	37.00	37.00	37.00	36.00	35.00	33.00	31.00	26.00	21.00	17.00	14.00	12.00	10.00
	67390.40	48115.70	34176.40	24049.40	16679.80	11319.20	7448.20	4729.00	2918.60	1801.20	1139.80	748.50	511.70	365.80
1.2	36.00	36.00	35.00	34.00	31.00	27.00	22.00	17.00	13.00	10.00	8.00	6.00	5.00	4.00
	60705.30	41525.80	27745.70	17893.80	10989.50	6362.20	3533.60	1948.50	1098.60	647.20	402.70	265.00	184.40	134.20
2	35.00	34.00	32.00	28.00	23.00	18.00	13.00	9.00	7.00	5.00	4.00	3.00	2.00	2.00
	52372.10	33546.50	20380.30	11523.60	6055.80	3048.90	1529.10	788.30	432.70	254.00	160.20	107.60	73.50	57.00
3	33.00	30.00	26.00	21.00	15.00	11.00	7.00	5.00	4.00	3.00	2.00	2.00	1.00	1.00
	42905.20	25035.30	13370.30	6548.00	3026.20	1378.10	644.80	325.50	176.90	104.70	68.10	43.60	32.70	25.70
5	28.00	23.00	17.00	12.00	7.00	5.00	3.00	2.00	2.00	1.00	1.00	0.00	0.00	0.00
	27520.40	13211.00	5697.70	2266.90	884.50	363.20	163.50	81.60	35.21	27.33	15.13	7.84	6.82	6.00
7	22.00	16.00	11.00	7.00	4.00	2.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00
	16988.90	6877.70	2483.10	846.90	300.80	114.35	47.55	24.64	8.20	6.50	5.10	3.90	2.80	2.00
10	16.00	10.00	6.00	3.00	2.00	1.00	0.00	0.00	0.00	0.00	-1.00	-1.00	-1.00	-1.00
	8010.10	2553.20	728.40	171.82	59.60	22.40	7.20	4.40	2.20	0.50	0.00	0.00	0.00	0.00
15	8.00	4.00	2.00	1.00	0.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	2124.30	462.50	92.90	15.70	4.70	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

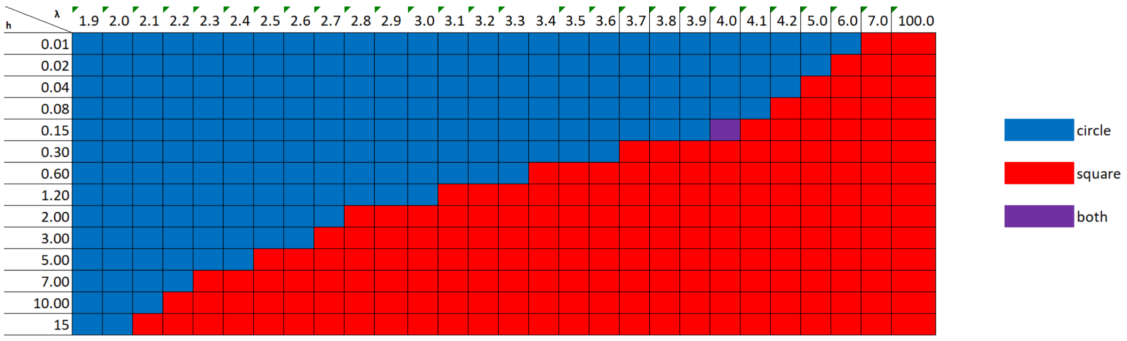
Table 5: The size of the least efficient square and the corresponding energy for $\lambda = 1.9$ to $\lambda = 3.2$

	3.3	3.4	3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	5.0	6.0	7.0	100.0
0.01	45.00	45.00	45.00	46.00	46.00	47.00	47.00	47.00	47.00	47.00	48.00	49.00	49.00	49.00
	3259.40	2886.30	2581.50	2338.00	2134.70	1964.00	1824.50	1704.60	1601.00	1510.70	1092.60	876.20	770.70	596.00
0.02	44.00	45.00	45.00	46.00	46.00	46.00	46.00	47.00	47.00	47.00	48.00	48.00	49.00	49.00
	3098.40	2720.70	2415.80	2165.00	1961.60	1490.60	1645.50	1524.20	1420.50	1330.20	904.40	680.80	574.70	400.00
0.04	44.00	44.00	44.00	44.50	45.00	45.00	45.00	45.00	45.00	45.00	41.00	33/34	30.00	24/25
	278106.00	2399.30	2088.40	1833.40	1624.80	1450.10	1302.70	1177.40	1070.10	977.60	552.50	371.30	298.80	200.00
0.08	41.00	41.00	40.50	40.00	38.00	36.00	34.00	32.00	30.00	28.00	20.00	17.00	15.00	12.00
	2189.30	1802.70	1491.30	1241.10	1040.40	880.50	754.60	653.30	576.70	513.70	274.60	184.80	148.90	100.00
0.15	33.00	30.00	27.00	24.00	21.50	20.00	18.00	17.00	16.00	15.00	11.00	9.00	8.00	6.00
	1372.70	1056.40	829.90	667.60	548.80	460.80	394.00	342.20	301.40	268.80	144.60	97.70	78.90	53.30
0.30	17.00	15.00	13.00	12.00	12.00	10.00	9.00	8.00	8.00	7.00	5.00	4.00	4.00	3.00
	644.20	491.20	385.90	311.20	256.90	216.70	186.60	162.80	143.70	129.00	70.70	48.20	38.60	26.60
0.60	8.00	7.00	6.00	5.00	5.00	4.00	4.00	4.00	3.00	3.00	2.00	2.00	2.00	1.00
	272.40	210.00	166.90	135.40	114.00	96.20	84.30	73.90	64.80	59.10	33.30	23.10	18.10	13.20
1.2	4.00	3.00	3.00	2.00	2.00	2.00	2.00	2.00	1.00	1.00	1.00	1.00	1.00	0.00
	100.30	80.40	64.10	53.20	45.90	39.60	34.30	29.50	26.00	24.20	15.00	9.60	6.90	5.60
2	2.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.00
	43.10	32.60	28.30	24.50	21.20	18.30	15.80	13.50	11.57	9.81	6.18	5.32	4.84	4.00
3	1.00	1.00	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	19.68	14.60	10.24	7.51	7.08	6.70	6.36	6.05	5.78	5.53	4.18	3.32	2.84	2.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-1.00	-1.00	-1.00
	5.20	4.50	4.00	3.50	3.10	2.70	2.40	2.00	1.80	1.60	0.20	0.00	0.00	0.00
7	0.00	0.00	0.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	1.20	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
15	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00	-1.00
	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 6: The size of the least efficient circle and the corresponding energy for $\lambda = 3.3$ to $\lambda = 100.0$

λ	h
1.9	15.0
2.0	15.0
2.1	10.0
2.2	7.0
2.3	7.0
2.4	5.0
2.5	3.0
2.6	3.0
2.7	2.0 (the two are equal for $h = 2.0$
2.8	1.2
2.9	1.2
3.0	1.2
3.1	0.6
3.2	0.6
3.3	0.6
3.4	0.3
3.5	0.3
3.6	0.3
3.7	0.15
3.8	0.15
3.9	0.15
4.0	0.15 (the two are equal for $h = 0.15$
4.1	0.08
4.2	0.04
5.0	0.02
6.0	0.01
7.0	Nowhere
100.0	Nowhere

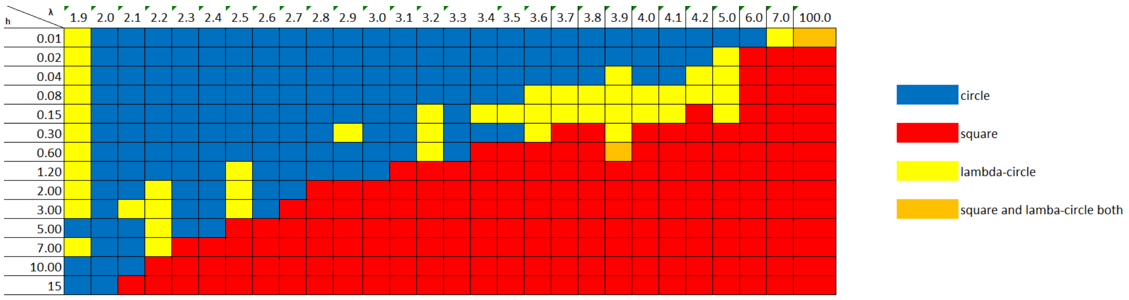
Table 7: Largest h for which the circle is more efficient than the square when looking at the maximum Hamiltonian



h \ λ	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3.0	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	5.0	6.0	7.0	100.0
0.01	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥
0.02	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥
0.04	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥
0.08	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥
0.15	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟪	🟥	🟥	🟥	🟥	🟥	🟥
0.30	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
0.60	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
1.20	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
2.00	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
3.00	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5.00	🟦	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
7.00	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10.00	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

🟦 circle
🟥 square
🟪 both

Figure 3: A coloured table illustrating the most efficient shape for values of λ and h, for the circle and the square.



h \ λ	1.9	2.0	2.1	2.2	2.3	2.4	2.5	2.6	2.7	2.8	2.9	3.0	3.1	3.2	3.3	3.4	3.5	3.6	3.7	3.8	3.9	4.0	4.1	4.2	5.0	6.0	7.0	100.0
0.01	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟧
0.02	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟥	🟥	🟥
0.04	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟦	🟦	🟨	🟨	🟥	🟥	🟥
0.08	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟥	🟥
0.15	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟦	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟨	🟥	🟨	🟥	🟥	🟥
0.30	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟦	🟦	🟨	🟦	🟦	🟦	🟨	🟥	🟥	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥
0.60	🟨	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟦	🟨	🟦	🟥	🟥	🟥	🟥	🟥	🟧	🟥	🟥	🟥	🟥	🟥	🟥	🟥
1.20	🟨	🟦	🟦	🟦	🟦	🟦	🟨	🟦	🟦	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
2.00	🟨	🟦	🟦	🟨	🟦	🟦	🟨	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
3.00	🟨	🟦	🟨	🟨	🟦	🟦	🟨	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
5.00	🟦	🟦	🟦	🟨	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
7.00	🟨	🟦	🟦	🟨	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
10.00	🟦	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥
15	🟦	🟦	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥

🟦	circle
🟥	square
🟨	lambda-circle
🟧	square and lamba-circle both

Figure 4: A coloured table illustrating the most efficient shape for values of λ and h, for the circle, the lambda-circle and the square.

mesoscopic regime. If the size of the critical radius is close to the size of the grid, the critical radius seems to increase as λ increases. If we are in the mesoscopic regime however, the size of the critical radius decreases for increasing λ. This seems to indicate that for an arbitrarily large lattice, the size of the critical radius should decrease for increasing λ.

We do not see any consistent region for which the λ-circle is the most efficient shape. We wanted to study this shape because it might combine the minimizing of long-range interactions of the circle with the maximizing of area per neighbouring vertices of the square. Since the λ-circle tends to the square as λ tends to ∞, it would also provide a continuity in the most efficient shape. We have however found consistent region of parameters for which it was the most efficient shape.

We also want to establish a relationship between the size of the critical droplet and h, J and λ. We will do this in chapter 4. To get an idea of the relationship between our parameters and the critical radii, we look at the surface plots for the square and the circle. These can be found in Figures 5 and 6. For these plots we have transformed the radii into diameters, since we use the diameters for the fits for the formula's for the critical diameters.

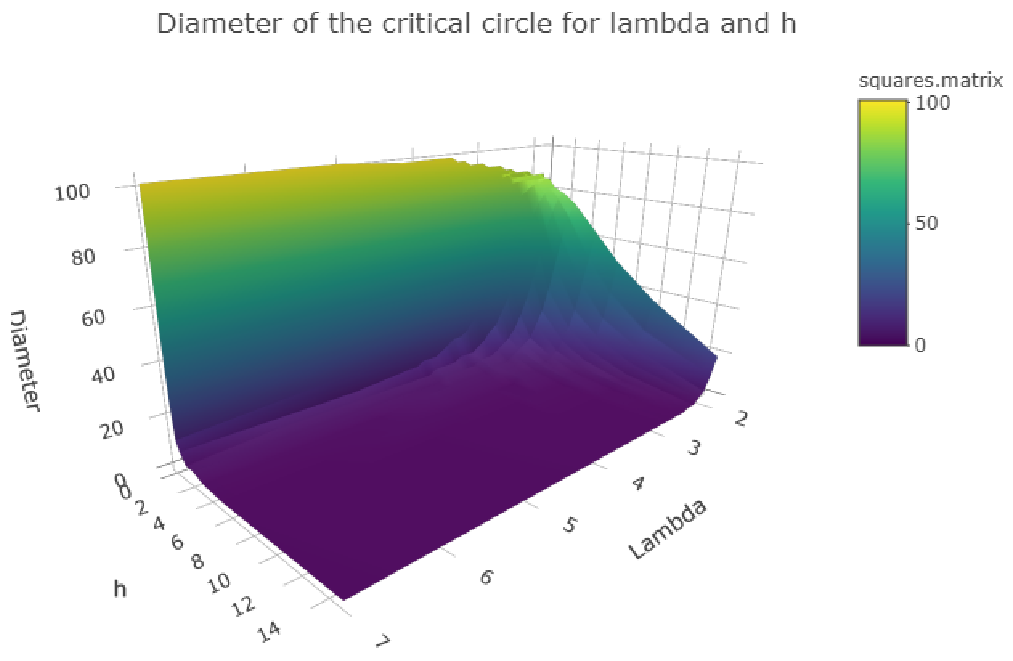


Figure 5: Diameter of the critical circle as a function of λ and h.

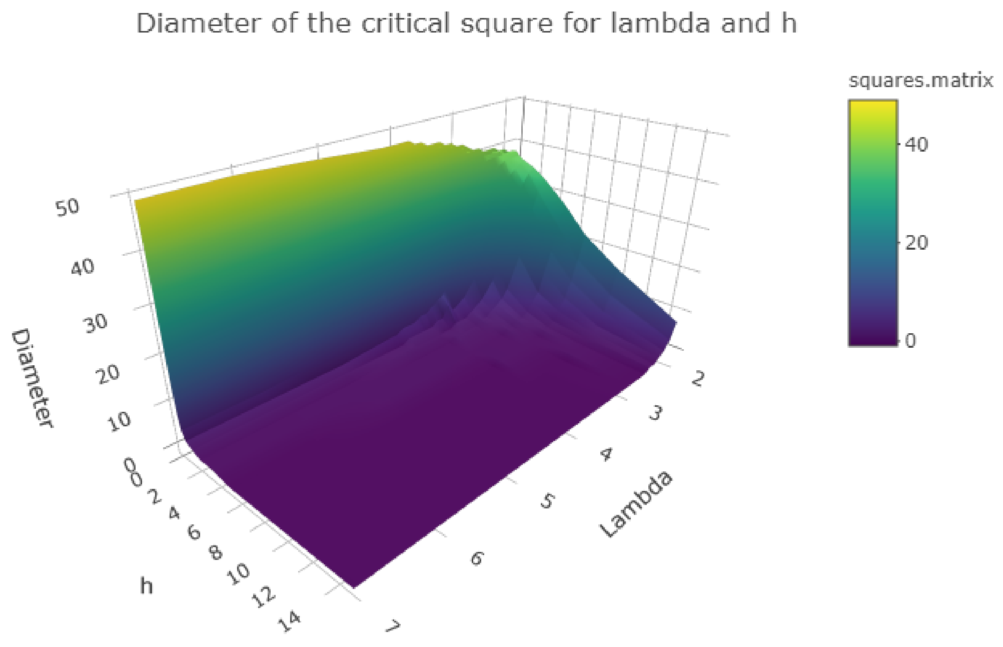


Figure 6: Diameter of the critical square as a function of λ and h.

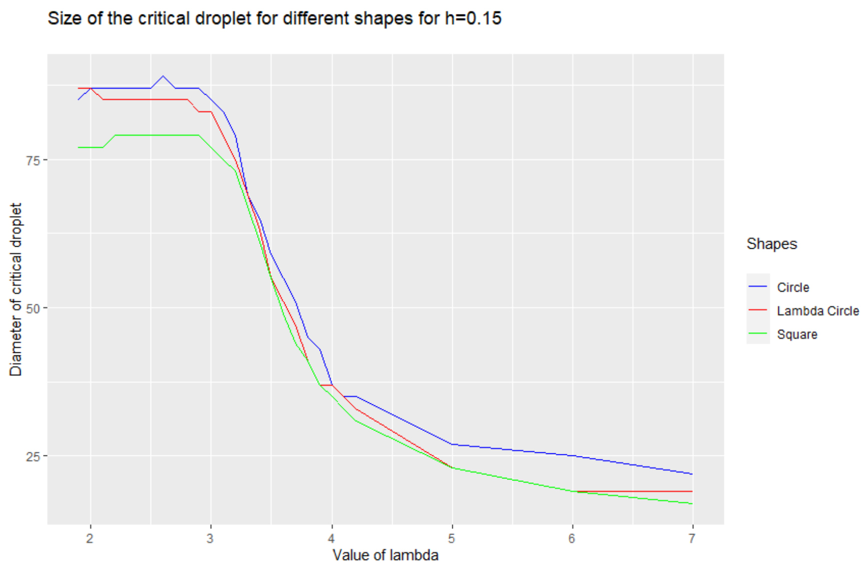


Figure 7: The size of the critical droplet for different shapes and different values of lambda for h=0.15

4 Size of the critical droplet

To look at the relationship between the size of the critical droplet and λ, we look at figures 7, 8 and 9. We suspect that this relationship is of the form $|d_{\text{critical}}| \sim \frac{1}{\lambda^c}$ for some c.

However, the graphs in these figures are not of this form. This is because for small λ, the size of the critical droplet is of the same order of magnitude as the size of the lattice, which influences the Hamiltonian. We are interested in results for arbitrarily large lattices, so we should look at results on the mesoscopic scale, where both the resolution of the lattice and the size of the lattice do not impact the positive component.

We want to look at several candidates for formulas for the size of the critical droplet and see how well they fit the sizes we found for the least efficient droplets for our shapes. We transform our radii into diameters for better fits, because a square (or circle) with radius 0 was not empty, but contained one vertice. This makes it difficult to properly fit exponential functions, so we choose to transform the data. Since a shape with radius 0 was not empty but still had the origin contained in it, we transform our radii into diameters by $d = 2r + 1$.

We fit our formulas to both the circle and the square, to see if there is any significant difference in how well they fit and the values which fit best.

For each h, we fit the curves on the values for λ from the minimum λ such that increasing λ by 0.1 changes the size of the critical droplet by 2 or more, to the maximum λ such that the critical droplet has at least diameter 1 and is not $-\mathbb{1}$ itself. As we discussed earlier, as the critical droplets gets too large, the size of the lattice starts to play a role. We want a formula for the size of the critical droplets for arbitrarily large lattices, and therefore we do not want to include data for which the size of our lattice is relevant. We filter away the data for which the critical droplet has diameter of less than 1, since the relevant range for our formula is for mesoscopic critical droplets. Since we only look at shapes with integer radii and so odd diameters, our data is much too coarse to fit a formula for these ranges. Our data is best at the mesoscopic scale, which is why we want to fit the formulas mostly on data for shapes on that mesoscopic scale.

We fit the formulas using the non-linear least squares function, nls, in R. This minimizes the squared error of the function compared to our data. The function fits our formulas to the data and gives us its most likely estimates for the constants and a residual standard error. The residual standard error is the positive square root of the mean of the square of residuals.

We fit both the data for the circle and the square to this formula, restricted as discussed above. We are mostly interested in the circle, since it is the more efficient shape in the region we are interested in, but we want to compare our outcomes to the square.

We start by fitting a power law to the data, a relatively simple formula, and we expect the size of the

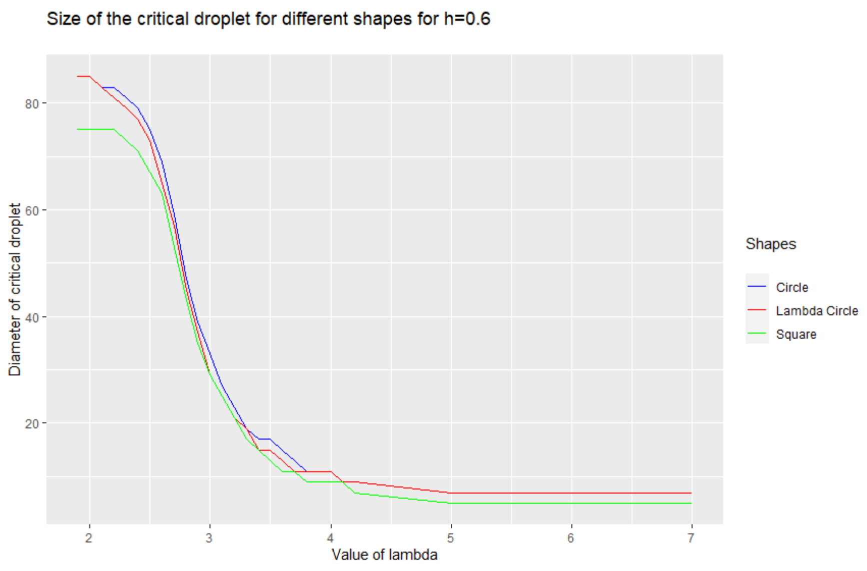


Figure 8: The size of the critical droplet for different shapes and different values of lambda for h=0.6

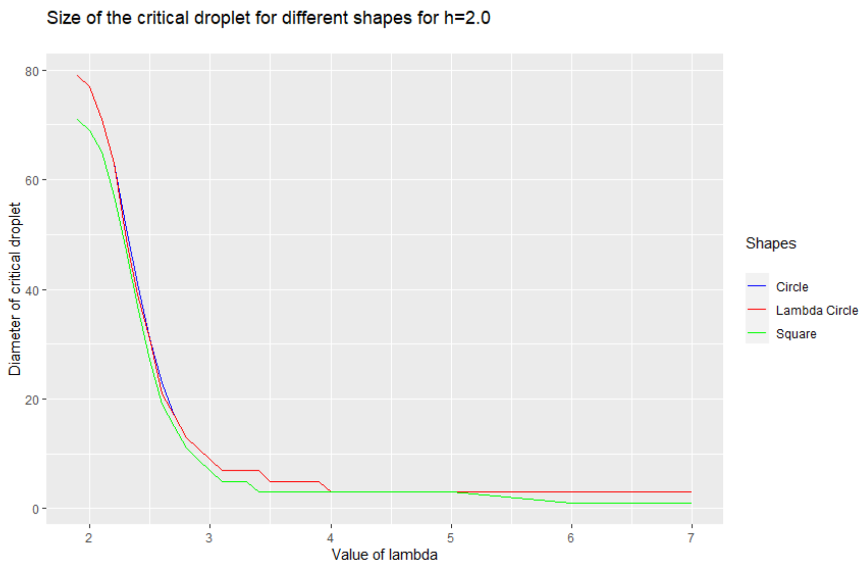


Figure 9: The size of the critical droplet for different shapes and different values of lambda for h=2.0

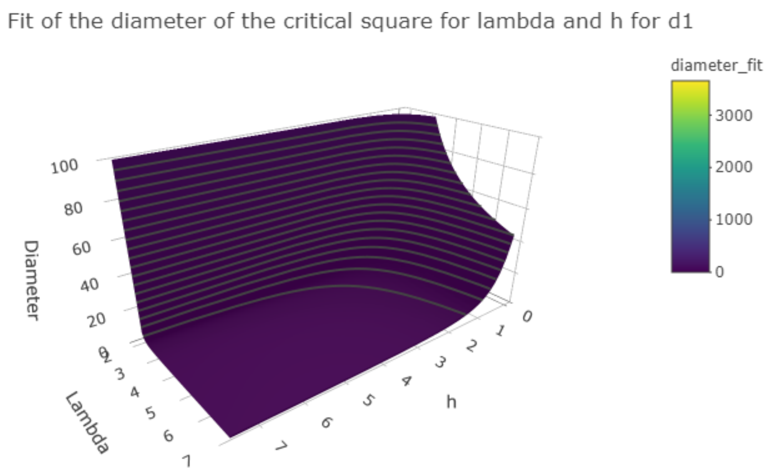


Figure 10: Best fit for our formula for d_{critical} in equation 8- relevant range

critical droplet to reduce exponentially based on the size of h and λ. Our formula is

$$d_{\text{critical}} \sim c * h^a * \lambda^b, \tag{8}$$

where we want to discover the most likely estimates for the values of c, a and b.

For the circle, we get

$$c = 2499.28, a = -0.81, b = -4.18.$$

Importantly, the residual standard error is 7.867. Since our diameters are between 80 and 3, this is a very large error, and indicates that this formula is not a good fit for the size. We can see a plot of the fit in figure 10. We can compare the plot to figure 5 and see quite some difference, especially in the exponential behaviour for small h.

For the square, we get

$$c = 1790.63, a = -0.78, b = -3.96$$

Our residual standard error is again very high at 9.22. For the square, this is not a good fit as well.

Our next guess is more intricate. When we look at the previous work done in the paper on the long range Ising model in one dimension [8], we see that proposition 3.3 in that paper gives a formula of the critical droplet as

$$\left(\frac{J}{h(\lambda-1)}\right)^{\frac{1}{\lambda-1}} \tag{9}$$

We use this formula for a more educated guess on the diameter of the critical droplet.

We look at

$$d_{\text{critical}} \sim \frac{c}{h^z} * h^{\frac{a}{\lambda-1}} \lambda^{\frac{b}{\lambda-1}}, \tag{10}$$

where we want to get a most likely estimate for the values of c, z, a, b. We have two factors for h, since we know that in the limit for $\lambda \to \infty$ our model should mimic the standard 2D Ising model, which has a critical droplet of size $\frac{2}{h}$. Since one of the factors is to a power of $\frac{a}{\lambda-1}$, this factor would go to 1 as λ tends to infinity. Therefore we need a second factor to allow the formula to go to the correct limit.

Setting the base in the λ-factor of the formula to λ instead of $(\lambda - 1)$ drastically reduces the error, which is why we choose the formula this way.

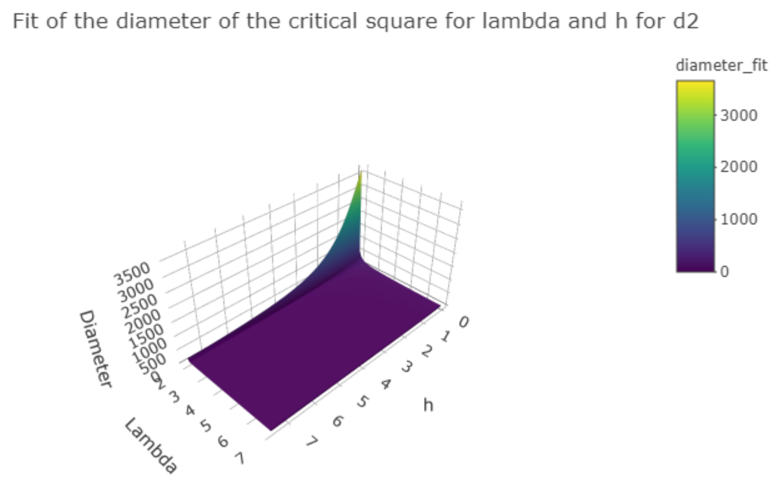


Figure 11: Best fit for our formula for d_{critical} in equation 10

If we look at the formula for 1D, equation 9, we suspect that we could use a factor $(\lambda - 2)$. However, we have data for $\lambda = 2$, which would set that term to zero, and we also have data for $\lambda = 1.9$. For both these values of λ, a formula with factors of $(\lambda - 2)$ would not work, while these values of λ are valid, and the long-range Hamiltonian still makes sense with $\lambda = 1.9$. The interaction between spins still decreases as the distance between them decreases, which is the important condition.

For the circle, we get

$$c = 0.019, a = 0.32, b = 12.77, z = 1.02$$

with a residual standard error of 6.0. This is a better estimate than before, but is still off by quite a bit. Moreover, the values are very counter-intuitive. b is positive, which differs from the formula in [8], and gives us a strange dependency on λ, where the diameter would increase for increasing λ, which is clearly at odds with our findings.

We have plotted the fitted function in figure 11, and a more relevant plot in figure 12. We see the major difference from the observed critical diameters in figure 5. Again, we see much steeper exponential behaviour for small h in our fit than in the observed data.

For the square, we get

$$c = 0.020, a = 0.34, b = 12.53, z = -1.01,$$

with a residual error of 7.44. This is a major error again. The values are very close to the values found for the circle, and have the same intuitive problems.

We try one last formula, in which we try to enforce the correct limit behaviour. We try to fit the formula

$$d_{\text{critical}} \sim \frac{2}{h}(ch^a\lambda^b)^{\frac{1}{\lambda-1}} \tag{11}$$

for the parameters a, b, c. We see that as λ tends to infinity, the limit is always right.

For the circle, we get as values

$$a = 0.16, b = -0.47, c = 114.02$$

with residual standard error of 7.85. This is still not a good fit, although the parameters make more intuitive sense than in the previous case. We see that the residual standard error is even larger than in the previous case, making this a worse fit. We can see the fit in the relevant range in Figure 13. Yet again, we see much steeper exponential behaviour for small h in our fit than in the observed data.

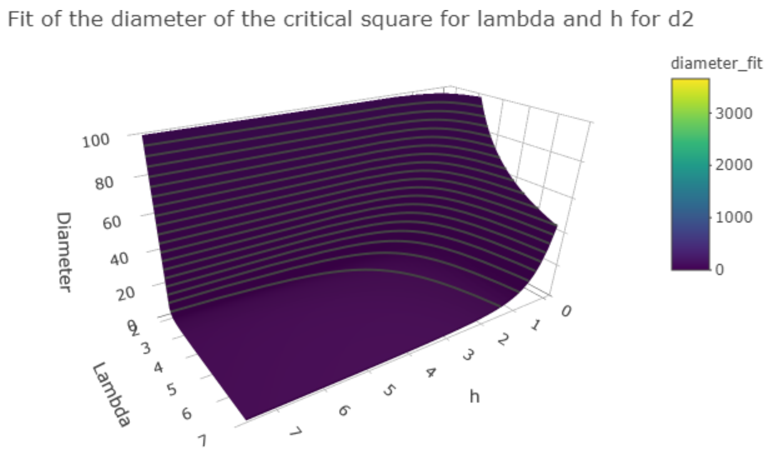


Figure 12: Best fit for our formula for d_{critical} in equation 10 - relevant range

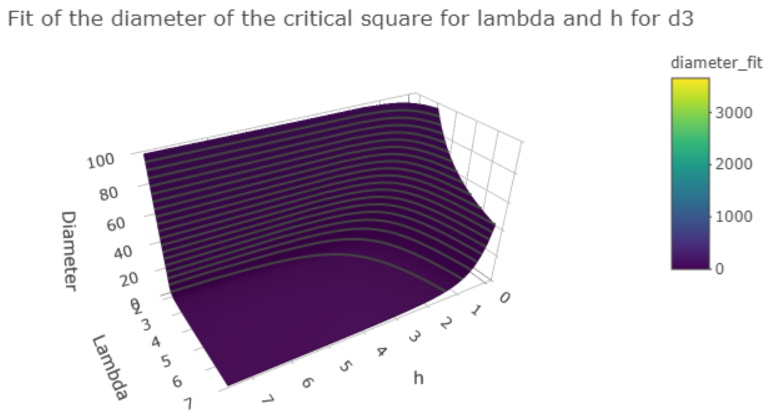


Figure 13: Best fit for our formula for d_{critical} in equation 11 - relevant range

Unfortunately, none of our guesses for the formula of the diameter of the critical droplet have been decent fits. There could be several reasons.

First of all, the cut-off point for our data might not have been chosen well. Perhaps the macroscopic effects of the lattice are still playing a large role in distorting the extremities of the data we have chosen to fit our formulas on, and we should have been more strict in the cut-off.

It is also possible that all or the majority of our data is affected by the lattice size. If this is the case, all of our data would be distorted, and the correct formula for large enough l might fit poorly on our set of data.

It could also be the case that the data is representative, and the formulas are not close to the 'true' formula for the critical droplet. In that case, we would simply need to guess a better formula, or need more analytical analysis of the Hamiltonian to provide a rigorous answer.

Furthermore, it could be that there is no general formula for all λ, and we need to look at different formulas for different ranges for λ. In the one-dimensional model, the formula for the critical droplet works for all $\lambda > 1$. This is both the minimum value of λ such that interactions at a longer-range get smaller and the dimension of the space. In our case, these two values are no longer the same, and perhaps we could have a different formula for $\lambda > 2$ than for $1 < \lambda \leq 2$. We note that the Hamiltonian is not summable as $L \to \infty$ for $\lambda \leq 2$

These objections could be researched by running simulations for very large L. This is computationally intensive, but could shed more light on the relationship between the critical droplet and h and λ.

5 Simulation for the critical droplet

Our hypothesis has been that either the square or the circle is the gateway shape. To test the hypothesis, we want to write an algorithm which simulates many runs from the metastable state to the global minimum and find the gateway shape for each of these runs.

We simulate these run under the assumption that the gateway shape is connected.

It is a difficult assumption to prove directly. Unfortunately, it is also a difficult assumption to prove inductively, since we would need to prove that for any connected shape the consecutive shape with the lowest Hamiltonian is one where the added spin is connected to the rest of the positive spin. It is however possible to construct counterexamples for this claim for certain lambda by taking non-convex connected shapes.

It is, however, an intuitively logical assumption, since the Hamiltonian increases if the distance between spins in the positive component increases. Therefore we would expect a convex connected shape as the gateway shape.

5.1 An efficient way to calculate ΔH

We want to rewrite the Hamiltonian such that it depends on the interaction between vertices inside the positive component and a general term which depends on the size of the positive component. This gives us a lot more room to increase the grid size as long as the size of the critical droplet remains small enough, since we only calculate interactions on the entire grid once at the beginning of the simulation.

For a given positive component R, we can write the Hamiltonian as

$$\begin{aligned}
H &= -\frac{J}{2}\sum_{i\notin R}\sum_{j\notin R, j\neq i}\frac{1}{||i-j||^\lambda} + J\sum_{i\notin R}\sum_{j\in R}\frac{1}{||i-j||^\lambda} - \frac{J}{2}\sum_{i\in R}\sum_{j\in R, j\neq i}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R| \\
&= -\frac{J}{2}\sum_{i\notin R}\sum_{j\neq i}\frac{1}{||i-j||^\lambda} + \frac{3J}{2}\sum_{i\notin R}\sum_{j\in R}\frac{1}{||i-j||^\lambda} - \frac{J}{2}\sum_{i\in R}\sum_{j\in R, j\neq i}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R| \\
&= -\frac{J}{2}\sum_{i\notin R}\sum_{j\neq i}\frac{1}{||i-j||^\lambda} + \frac{3J}{2}\sum_{j\in R}\sum_{i\notin R}\frac{1}{||i-j||^\lambda} - \frac{J}{2}\sum_{i\in R}\sum_{j\in R, j\neq i}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R| \\
&= -\frac{J}{2}\sum_{i\notin R}\sum_{j\neq i}\frac{1}{||i-j||^\lambda} + \frac{3J}{2}\sum_{j\in R}\sum_{i\notin R}\frac{1}{||i-j||^\lambda} - \frac{J}{2}\sum_{i\in R}\sum_{j\in R, j\neq i}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R| \\
&= -\frac{J}{2}\sum_{i\notin R}\sum_{j\neq i}\frac{1}{||i-j||^\lambda} + \frac{3J}{2}\sum_{j\in R}\sum_{i\neq j}\frac{1}{||i-j||^\lambda} - \frac{3J}{2}\sum_{j\in R}\sum_{i\in R, i\neq j}\frac{1}{||i-j||^\lambda} - \frac{J}{2}\sum_{i\in R}\sum_{j\in R, j\neq i}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R| \\
&=
\end{aligned} \tag{12}$$

Because of periodic boundary conditions, for any i the sum $\sum_{j\neq i}\frac{1}{||i-j||^\lambda}$ is equal. Say $H^{(0)} = \sum_{j\neq i}\frac{1}{||i-j||^\lambda}$. This means we can reduce our equation to

$$\begin{aligned}
H &= -\frac{J}{2}|\Lambda\backslash R|H^{(0)} + \frac{3J}{2}|R|H^{(0)} - 2J\sum_{j\in R}\sum_{i\in R, i\neq j}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R| \\
&= \frac{-J}{2}(|\Lambda\backslash|R| - 3|R|)H^{(0)} - 2J\sum_{j\in R}\sum_{i\in R, i\neq j}\frac{1}{||i-j||^\lambda} + h|\Lambda| - 2h|R|
\end{aligned} \tag{13}$$

Now say we change one element i from $\sigma_i = -1$ to $\sigma_i = +1$. Then we get

$$\Delta H = 2JH^{(0)} - 2h - 4J\sum_{j\in R, j\neq i}\frac{1}{||j-i||^\lambda} \tag{14}$$

5.2 The algorithm

We start with the negative state. Before our simulation, we select a maximum size for the positive component. We use our previous numerical results to select a size large enough to contain the critical shape.

We look at the energy difference with $H(-\mathbb{1})$. We recall that $E(\sigma) = H(\sigma) - H(-\mathbb{1})$. Therefore, for our starting state $-\mathbb{1}$ we set $E = 0$. In the first step, we add the origin of our lattice. We ran the simulations for a lattice of size 101×101. We then calculate ΔE using equation 14. During the runs of the simulation, we keep track of the positive component and a set of the elements of the lattice which border the positive component.

We use a recursive algorithm to continuously add new vertices to the positive component until the component reaches its determined maximum size.

For a given positive component, the algorithm looks at the vertices bordering the component and how many neighbours they have. If there exists a border vertice with 3 neighbours, it adds this to the component and repeats. If there exists no such number, it randomly selects a vertice from the border to add to the positive component. With probability 0.66 it selects a uniformly random border vertice with two neighbours, and it uniformly randomly selects a border vertice with one neighbour with probability 0.34. If the category it selects does not exist, it adds a random vertice from the other category.

The algorithm then calculates ΔE for the vertex it has selected to flip, and adds this to the current Hamiltonian. The algorithm then adds the selected vertex and the updated Hamiltonian to a list for the current simulation. The eventual output of the algorithm is a list of lists which consist of coordinates for vertice and the Hamiltonian after they were flipped from negative to positive.

5.2.1 Considerations

The algorithm had to be able to travel using logical paths to decrease computing time, but we still had to be able to reach the shapes we were interested in. Making an algorithm which only considered connectivity was not efficient enough. If only connectivity was considered, we would create shapes with holes or long arms of small width. Both of these shapes are unlikely to be the most efficient shape.

Therefore, we decided to always add vertices with three neighbours to the positive component before looking at vertices with less neighbours. This prevents the algorithm from looking at positive component with holes, and increases the convexity of the shapes it looks at, hopefully allowing it to look at more relevant shapes for each simulation, and therefore be more efficient.

However, we could not deterministically add all vertices with two neighbours before adding vertices with one neighbour. This would not allow us to look at the circle, and instead only create quasi-rectangles of increasing sizes.

This is why we chose a random method which looked at both vertices with two and one neighbour. This allows the algorithm to visit both circles and rectangles, which are our initial guesses for possible shapes of the critical droplet.

5.3 Results

First, we want to see if our algorithm works at all. For small l, we choose relatively high h, λ and see if we get a sensible shape. For $l = 20, h = 2.0, \lambda = 2.5$ and with 100000 simulations, we get Figure 14.

We note that this critical droplet is the semi-rectangle which one would expect for the short-range Ising model. This is an encouraging result.

We increase the number of simulation to get results for 200000 simulations total for the usual grid of 101×101.

For $h = 0.6, \lambda = 2.9$, we get the critical droplet depicted in 15.

From our simulations, we can also draw a graph for the minimal E for each of size of the positive component. This graph is in figure 17. We see that it is not yet clear that the figure that our algorithm spews out as the critical droplet is close to the actual critical droplet, since we do not see the Hamiltonian go down yet for the size we chose.

We can compare this graph to the results from our calculations earlier for the same values for λ and h. This graph can be found in figure 16. We notice that these graphs match fairly well.

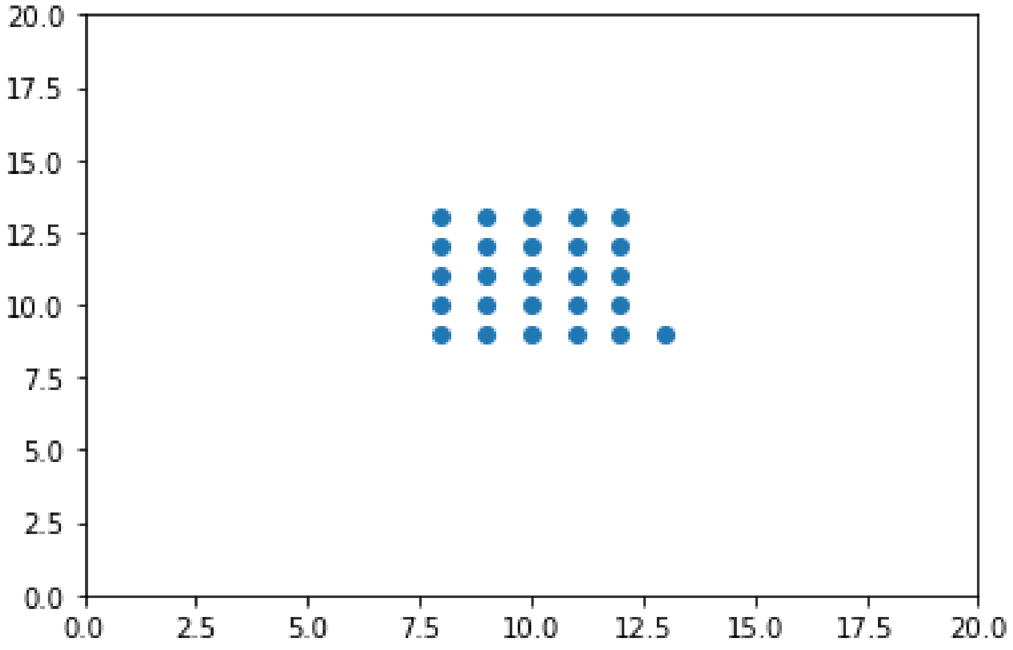


Figure 14: Critical droplet for $l = 20, h = 2.0$ and $\lambda = 2.5$

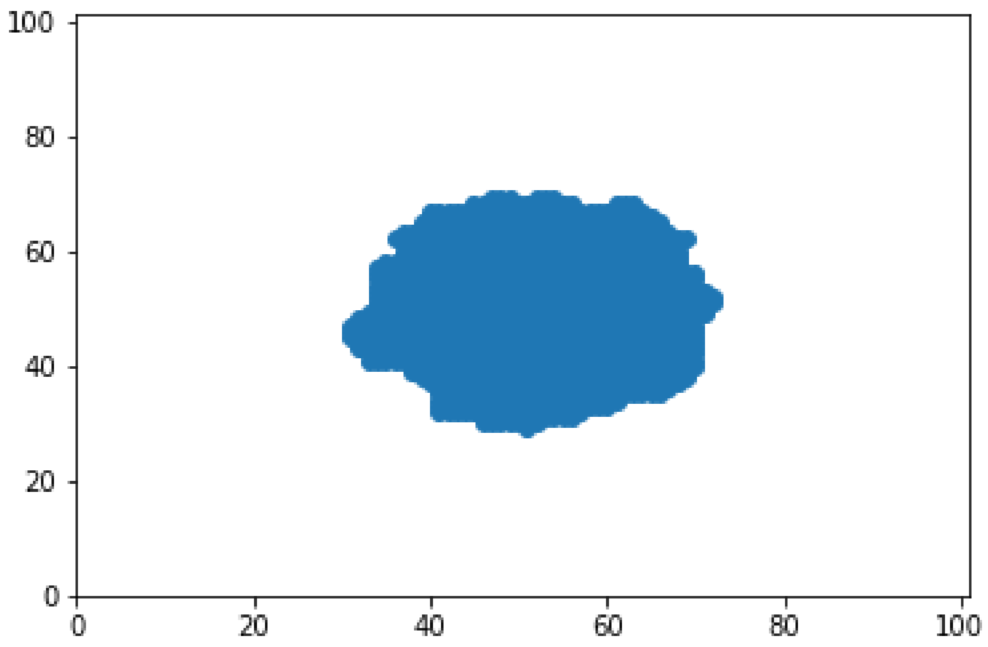


Figure 15: The resulting critical droplet from our simulations for $\lambda = 2.9, h = 0.6$ for 200000 simulations.

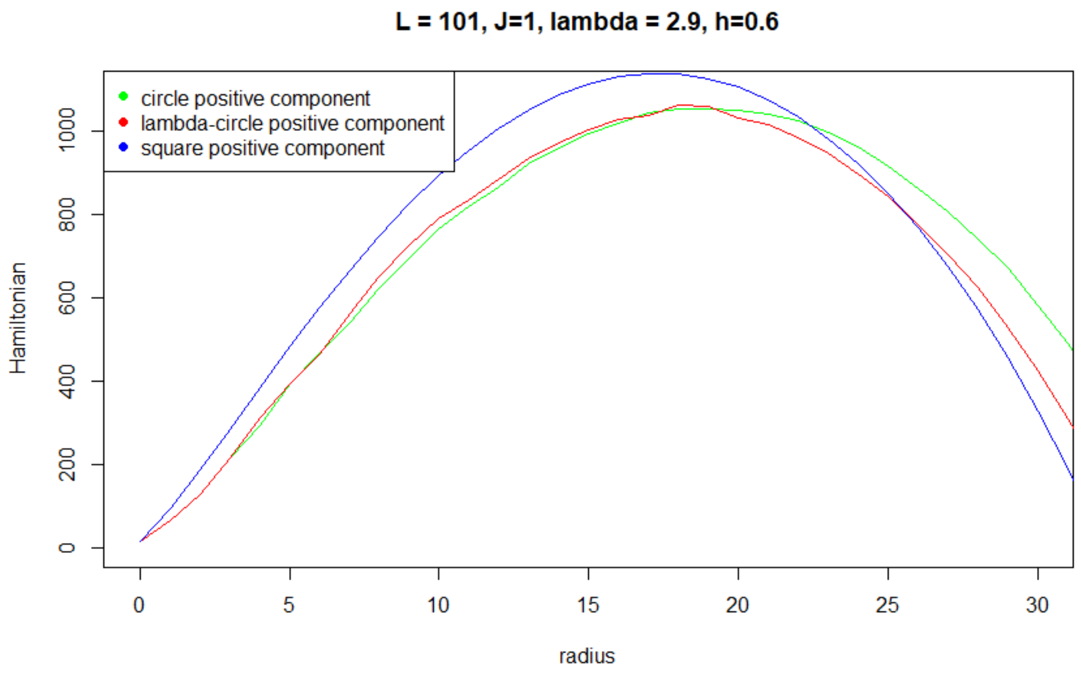


Figure 16: Hamiltonian for different radii for $\lambda = 2.9, h = 0.6$ and $J = 1$

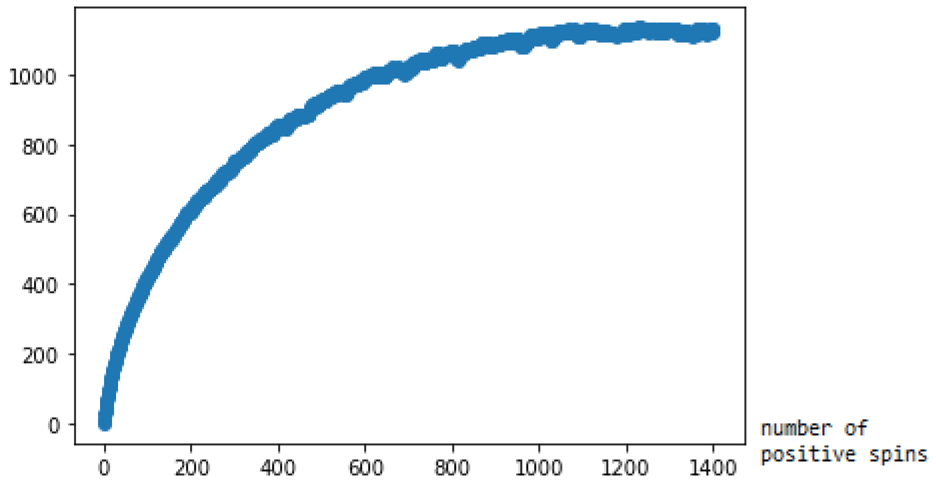


Figure 17: The resulting graph for the Hamiltonian from our simulations for $\lambda = 2.9, h = 0.6$ for 200000 simulations.

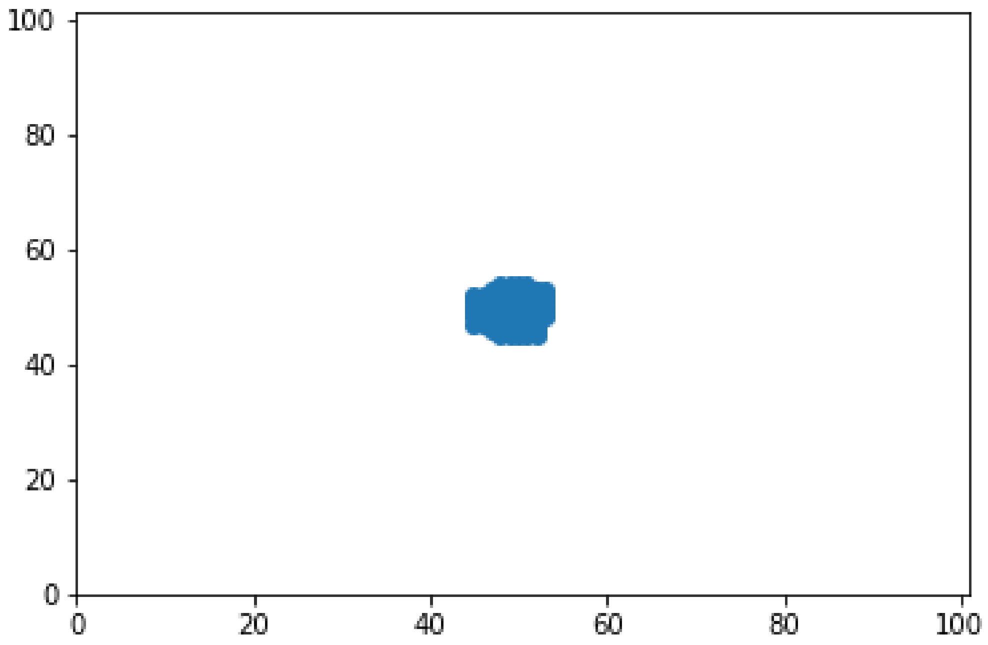


Figure 18: The resulting critical droplet from our simulations for $\lambda = 2.9, h = 2.0$ for 200000 simulations.

We ran the same simulations for $\lambda = 2.9, h = 2.0$, again with 200000 simulations. The results are in figures 18 and 19.

We ran the simulations again for $\lambda = 2.9, h = 2.0$, again with 200000 simulations. The results are in figures 20 and 21.

We notice that the the observed critical droplets are very close to what we would expect from our earlier work in chapter 3 in Figure 3. For $\lambda = 2.9, h = 0.6$ we get a mostly circular critical droplet, which is what we would expect based on Figure 3. Similarly, our result for $\lambda = 2.9, h = 2.0$ is less circular and more square-shaped, although it is not a perfect square. This also agrees with the results in Figure 3.

This indicates that our results in chapter 3 are very relevant to the shape of the critical droplet.

However, we do not know how quickly the algorithm converges in probability towards the critical droplet. The most efficient droplet we found is a random outcome of the simulations, and we do not exactly know how the distance towards the actual most efficient droplet (in number of spins on which the two differ) is distributed. Further research could go into estimating this distribution and more robustly estimating the representativeness of the algorithm.

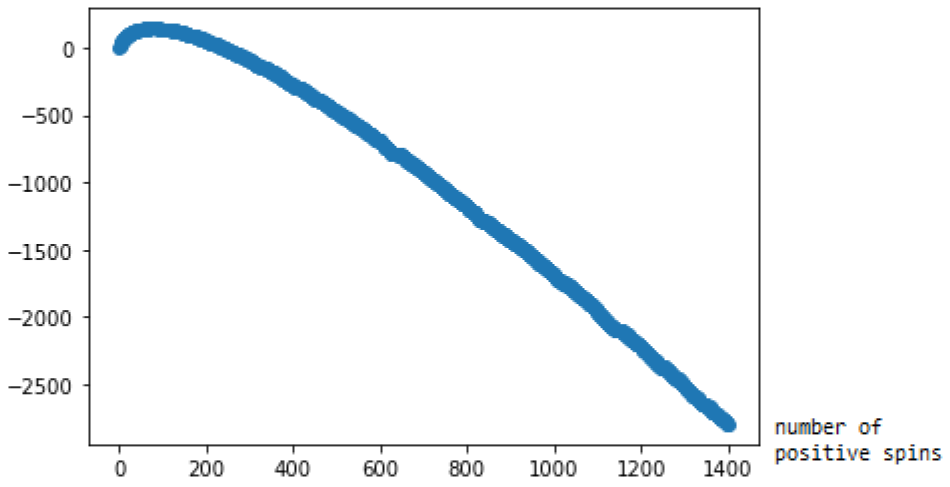


Figure 19: The resulting graph for the Hamiltonian from our simulations for $\lambda = 2.9, h = 2.0$ for 200000 simulations.

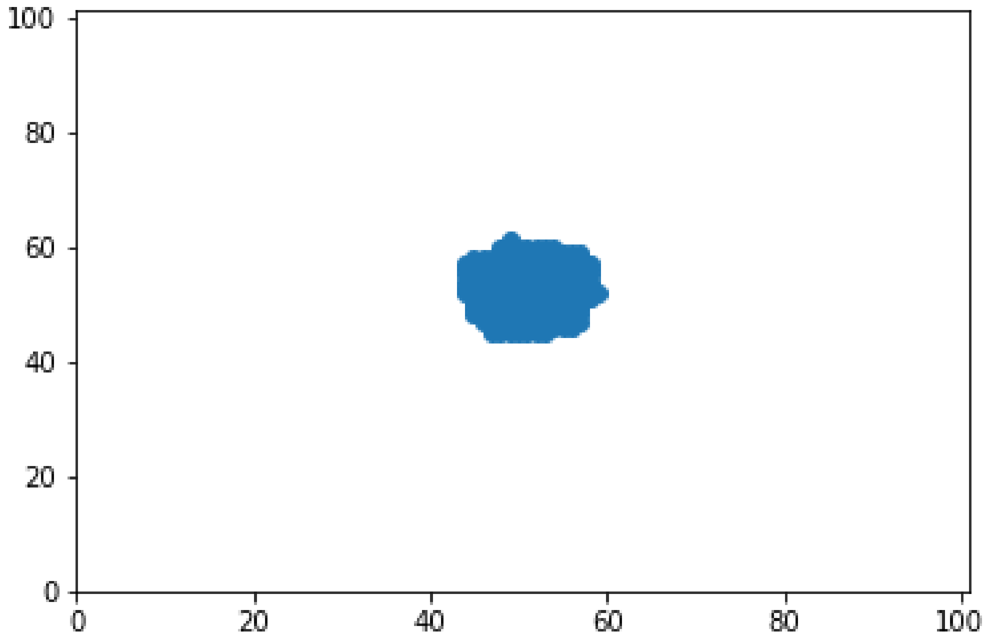


Figure 20: The resulting critical droplet from our simulations for $\lambda = 2.7, h = 2.0$ for 200000 simulations.

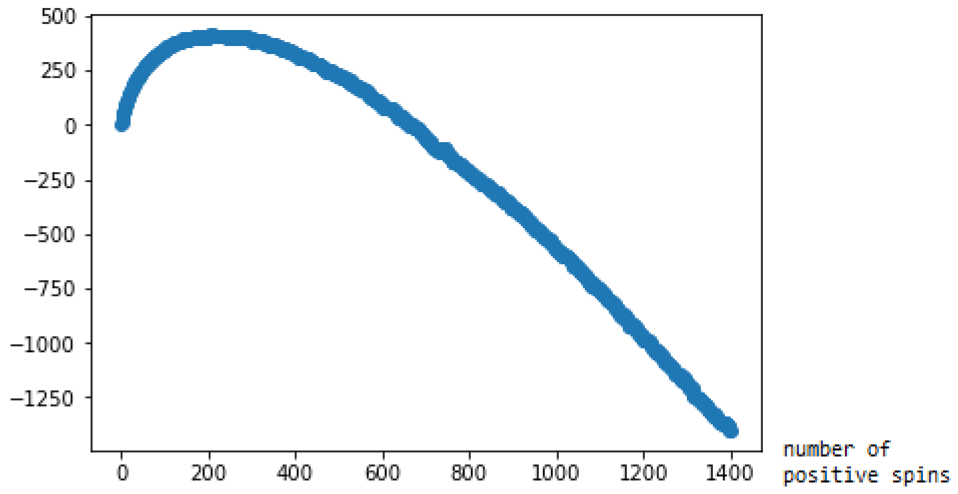


Figure 21: The resulting graph for the Hamiltonian from our simulations for $\lambda = 2.7, h = 2.0$ for 200000 simulations.

6 Metropolis-Glauber dynamics

As we mentioned in the introduction, we are interested in the distribution of the hitting time of $+\mathbb{1}$ when starting from the state $-\mathbb{1}$. For this, we use Metropolis-Glauber dynamics.

As we mentioned in the introduction, the Metropolis-Glauber algorithm is a Markov chain. At every step of time, we select a random vertice in our lattice Λ. We look at the difference in Hamiltonian if we flip the spin on this vertice. If $\Delta H < 0$, we always flip the spin on the vertice.

If $\Delta H > 0$, we flip the spin on the vertice with probability $e^{-\beta \Delta H}$.

In our algorithm in section C, we use the same way of calculating ΔH as in our algorithm for the optimal shape. This would allow us to apply the Metropolis-Glauber algorithm for larger lattices.

To look at the time it takes to get to the stable state from our metastable state, we look at the [5]. As we discussed in section 2.1.2, we look at the following theorem:

Theorem 1.. For any metastable state η,

$$\lim_{\beta \to \infty} \frac{1}{\beta} \log \mathbb{E} \tau^{\eta}_{\mathscr{X}^s} = \Gamma \tag{15}$$

We use this theorem under the assumption that conjecture 2.1.2 is true. The theorem would then give us that $\lim_{\beta \to \infty} \frac{1}{\beta} \log(\mathbb{E}[\tau^{-\mathbb{1}}_{+\mathbb{1}}]) = \Gamma$. We want to use this theorem to estimate Γ from the observed hitting times. An issue is the expected size of our Γ.

We can see in tables 3.1.1 and 3.1.1 that for $2 \leq \lambda \leq 3$, our Γ is at least 100 and for a large range of values, it is at least 1000.

For $\beta = 2.0$, this formula would indicate that the expected hitting time is in the order of 10^{86}, which is an unrealistically long time for our algorithm to run. Our algorithm can handle times up to $10^{1}0$, which would allow us to handle β up to 0.3.

We want to run the Markov-Chain for several values of β for constant values of and h. For these values ofλ, h we choose $h = 2.0$, $\lambda = 3.1$. This is a range with low enough that we can run the simulation for several values of β, but large enough critical diameters that we are still in the mesoscopic range. We run the Metropolis-Glauber algorithm for several β in increments of 0.05. We start at $\beta = 0.20$.

We run the Metropolis-Glauber algorithm until 90% of the lattice is positive. This reduces the time the algorithm needs to run without fundamentally altering the dynamics of the model, so that our theorem is

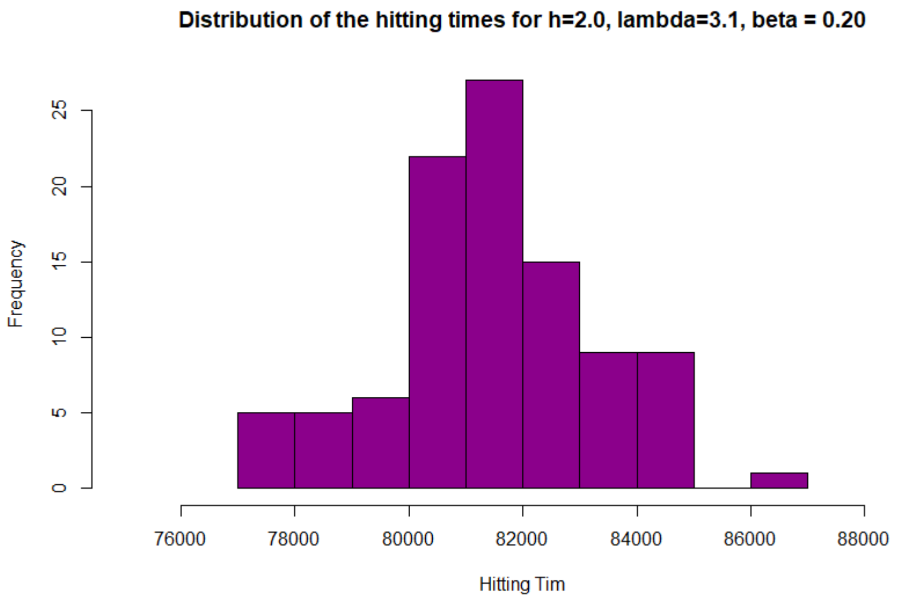


Figure 22: Hitting times for $h = 2.0, \lambda = 3.1, \beta = 0.20$

β	Mean Hitting Time	p-value
0.20	81436.61	$\sim 10^{-16}$
0.25	154293.5	$\sim 10^{-15}$
0.30	622286.2	$\sim 10^{-15}$
0.35	12799798	0.4898

Table 8: Caption

still applicable. Once the critical droplet has been hit, the remaining time to get to $+\mathbb{1}$ should be of order $o(1)$ as discussed in our definition of metastability for β large enough.

We repeat the algorithm 100 times for the first values of β for which this is computationally doable. For larger values of β we reduce this number in order to get a manageable running time.

Our code outputs the hitting times for the runs of the Metropolis-Glauber. First of all, we try to see if the hitting times are exponentially distributed. We do this by fitting an exponential distribution to our data with a most likely estimate, and using the Kolmogorov-Smirnov test to see how well the estimated exponential distribution actually fits the data. This Kolmogorov-Smirnov test gives us a p-value. If this p-value is under α (usually 0.05), we can reject the hypothesis that our data is exponentially distributed. Secondly, we look at the mean of the hitting times, and we try to compare the means of the hitting times to look at their relationship to β.

For $\beta = 0.20$, our mean for the hitting time is 81436.61. We can see the distribution in figure 22. This does not look exponentially distributed. The p-value of our Kolmogorov-Smirnov test is of the order 10^{-16}, so we can decisively say that the hitting times are not exponentially distributed for this choice of β.

For further choice of β, the results are summarized the result in table 8. Graphs of the empirical distribution of the hitting times are in figures 23, 24 and 25. Since these are empirical observations which are results of 100 observations of the hitting time which is a random variable, they are not perfectly representative of the true distribution of the hitting time. For example, for $\beta = 0.20$ in graph 22 we notice an empty bin before the last non-empty bin. This does not mean a 0 percent chance of hitting times within this interval for the true distribution, but rather that in our 100 observations of the hitting time we have not found one within the values for this bin.

We see that for $\beta = 0.35$ we do get an exponential distribution. Both the graph in figure 25 and the

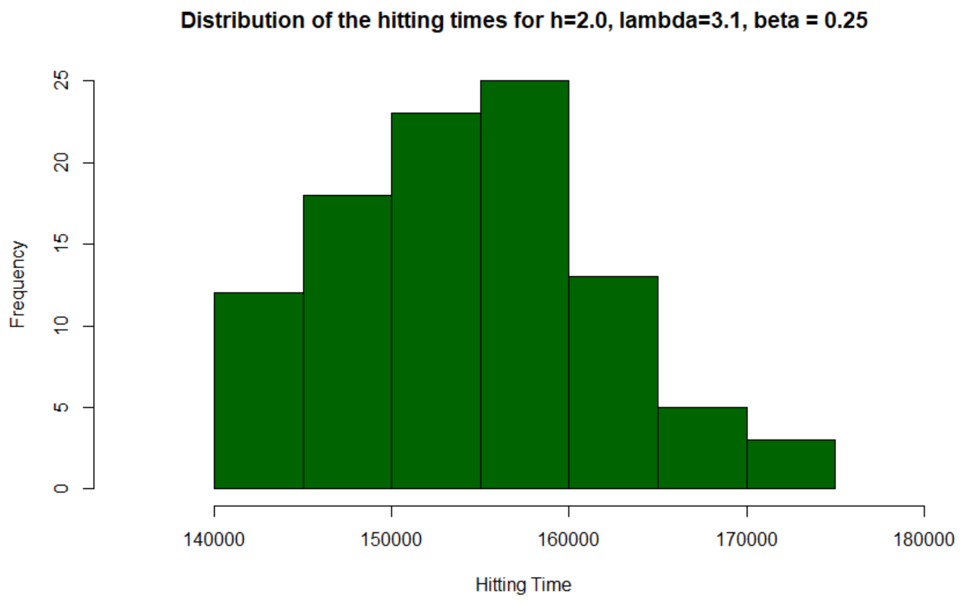


Figure 23: Hitting times for $h = 2.0, \lambda = 3.1, \beta = 0.25$

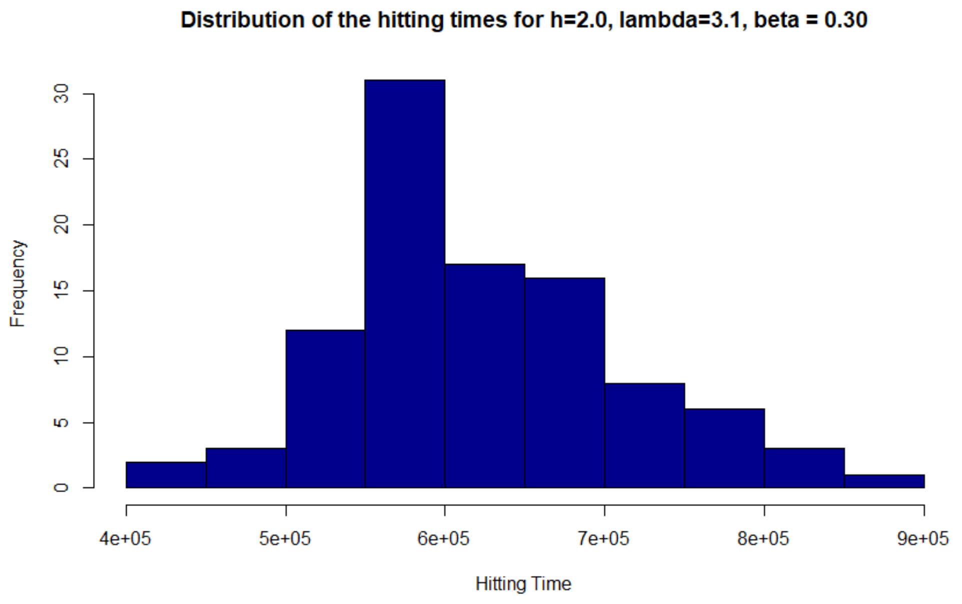


Figure 24: Hitting times for $h = 2.0, \lambda = 3.1, \beta = 0.30$

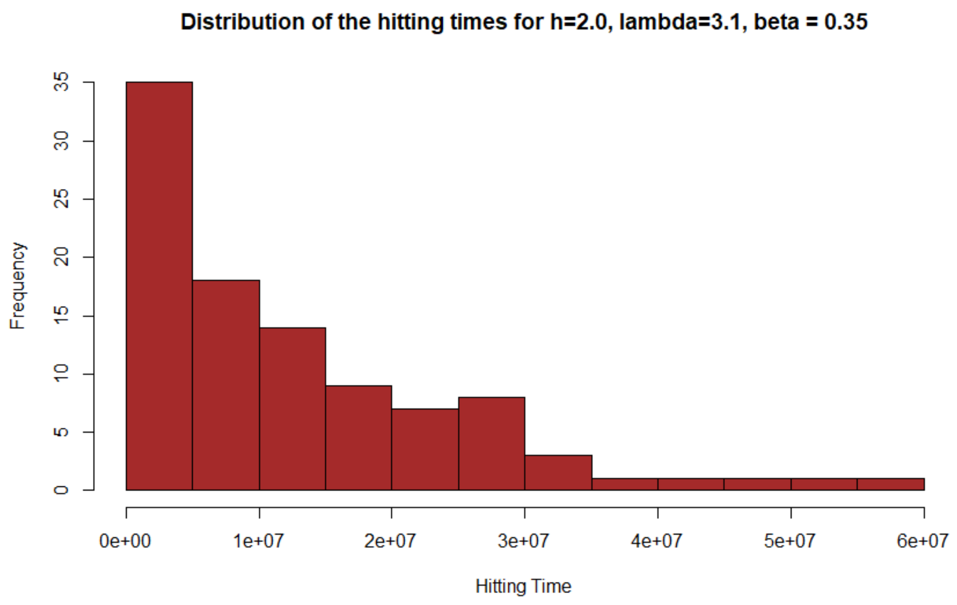


Figure 25: Hitting times for $h = 2.0, \lambda = 3.1, \beta = 0.35$

Kolmogorov-Smirnov test confirm that the data is exponentially distributed.

In table 3.1.1 we see that our Γ is around 73.50. Since our hitting time is expected to scale with $e^{\beta\Gamma}$, this would mean that a 0.05 increase in β would mean a factor of 40 difference in hitting time.

We cannot observe this factor yet. We notice that since for $\beta = 0.20, 0.25, 0.30$ our data was not yet exponentially distributed, we are not yet in the regime where the behaviour of the hitting time is as described in theorem 6 in the limit. Therefore we can not use the observed hitting times to estimate β for this combination of parameters.

7 Discussion

7.1 Results

We found that for certain regimes of λ, h, the circle is the shape with the lower maximum Hamiltonian in its path from $-\mathbb{1}$ to $+\mathbb{1}$. There were some difficulties with finding a large enough mesoscopic range, as discussed in section 7.1.1. We do, however, have good results on the range of parameters for which the circle is more efficient.

The circle is more efficient on a large range of lower λ and h values, which is what we expected to find. This is an important difference to the short-range Ising model, which always has square-shaped critical droplets. Unfortunately, we were not able to use this data to get a good hypothesis for a formula for the diameter of the critical droplet. Our educated guesses still had a large error, and were not good fits.

The algorithm to simulate paths in order to find the critical droplet agreed with our earlier findings on the square vs the circle. Assessing its reliability in approaching the critical droplet is difficult however, although the results are promising. Work could be done on assessing its convergence in probability and computational efficiency to get more rigid results.

We saw that the Metropolis-Glauber dynamics's expected hitting times were indeed exponential for large enough β. Because of the time consumption of running the Metropolis-Glauber algorithm for large β, we tested the dynamics for relatively low β. Therefore, we were not yet in the asymptotic regime and we were not able to use theorem 6 to estimate Γ.

A final unfortunate computer crash interrupted the Metropolis-Glauber algorithm for $\beta = 0.4$. Since the expected running time of the simulation for this value was a week, there was no time to run the simulation again.

7.1.1 Computational difficulties

With most of the calculations and simulations of the long-range 2D Ising model, we ran into the computational intensity of the model compared to the short-range 2D Ising model. Because every vertice interacts with every other vertice in the lattice, most computations scale poorly with size. For the short-range Ising model, the total number of interactions on an $L \times L$-sized lattice is $2L^2$, since each vertex only interacts with its 4 neighbours. However, with the long-range Ising model, the total number of interactions on an $L \times L$-sized lattice is $\frac{L^4 - L^2}{2}$. This scales much worse for larger lattices, although there are some ways to make the simulations more efficient. We do not have to calculate each interaction anew for each new lattice configuration, there are several ways to reduce the total number of calculations needed per lattice configuration. Some aspect of the lack of scalability remained, and hampered the ability of simulations to get rigid results.

When investigating the differences between the circle and the square, computational limits kept the size of the lattice at 101×101, which narrowed the range of parameters for which the resulting critical droplet was mesoscopic. Because of this, the data needed to estimate a formula for the radius of the critical droplet was hard to process.

When running simulations to find the critical droplet, the computational intensity of calculating the Hamiltonian for each state in a simulation limited the amount of simulations to 200000, and limited the amount of parameters for which simulations could be ran.

Similarly, the exponential time-scaling of the Metropolis-Glauber algorithm combined with the computational difficulties of the long-range Ising model heavily restricted the choice of β and the amount of simulations which could be run. Therefore, although our results confirm the exponential distribution for high enough β, the result is not very conclusive.

7.2 Future directions

In order prove rigorously that the critical droplet is a circle, one could try using Riemann integration to calculate bounds for the Hamiltonian for the square and the circle.

Using these bounds, one could also calculate the exact regions for which the circle or square is the most efficient shape. Importantly, if the value for ΔH between two shapes with the same number of positive spins has a lower bound, one could use the calculated bounds for the Hamiltonian for the circle or square to definitively prove that they are the most efficient shape.

One could also look at the behaviour of the system for smaller grid-resolutions. In our system, the vertices were on a square grid at fixed distance 1 from each other. However, by appropriately scaling h, J and L one could study the behaviour of the system as the grid gets ‘finer’ and look at the limit to a continuous space. In this continuous case, one would expect the circle as the solution to the long-range isoperimetric problem, one could look if this limit indeed holds.

Finally, getting a result on the size of the critical droplet is an important result for future directions. In chapter 4 we already discussed some of the reasons why our fits were not successful. Calculations on a larger lattice could increase the range on which one could fit possible formulas. Similarly, Riemann integral analysis of the Hamiltonian for the square and the circle might result in a rigorous formula for the size.

A Code for calculating the Hamiltonian for the circle, the square and the λ-circle

```
#include <iostream>
#include <cmath>
#include <fstream>
#include <math.h>
#include <windows.h>
#include <vector>
#include <array>

using namespace std;

int** statewithsquarecomponent(int length, double radius)
    {
      int** array2D = 0;
      array2D = new int*[2*length+1];

      for (int h = 0; h < 2*length+1; h++)
      {
            array2D[h] = new int[2*length+1];

            for (int w = 0; w < 2*length+1; w++)
            {

                if (abs(h-length)<= radius && abs(w-length) <= radius){
                    array2D[h][w] = 1;
                } else{
                    array2D[h][w] = -1;
                }
            }
      }

      return array2D;
    }

int** statewithcirclecomponent(int length, double radius)
    {
      int** array2D = 0;
      array2D = new int*[2*length+1];

      for (int h = 0; h < 2*length+1; h++)
      {
            array2D[h] = new int[2*length+1];

            for (int w = 0; w < 2*length+1; w++)
            {
                  // fill in some initial values
                  // (filling in zeros would be more logic, but this is just for the
                      ↪ example)

                if (sqrt(pow((h-length),2) + pow((w-length),2)) <= radius) {
                    array2D[h][w] = 1;
                } else{
                    array2D[h][w] = -1;
                }
            }
      }

      return array2D;
    }

int** statewithlambdacomponent(int length, double radius, double lambda)
    {
      int** array2D = 0;
      array2D = new int*[2*length+1];

      for (int h = 0; h < 2*length+1; h++)
      {
            array2D[h] = new int[2*length+1];

            for (int w = 0; w < 2*length+1; w++)
            {
                  // fill in some initial values
                  // (filling in zeros would be more logic, but this is just for the
                      ↪ example)

                if (pow(pow(abs(h-length),lambda) + pow(abs(w-length),lambda), 1/lambda) <=
                    ↪  radius) {
                    array2D[h][w] = 1;
                } else{
                    array2D[h][w] = -1;
                }
            }
      }

      return array2D;
    }

double Hamiltonian(int **array, double J, double lambda, double h)
{
    int length=101;
    long double H = 0.0;
    for(int a = 0; a < length; a++){
        for(int b = 0; b < length; b++){
            for (int c = 0; c < length; c++){
                for(int d = 0; d < length; d++){
                    if (a != c || b != d){
                        H = H - (0.5*J*array[a][b]*array[c][d])/(pow(pow(min(abs(a-c), min(
                            ↪ abs(a-c+length), abs(a-c-length))),2)+ pow(min(abs(b-d), min(
                            ↪ abs(b-d+length), abs(b-d-length))),2),(lambda/2)));
                    }
                }
            }
           H = H-h*array[a][b];
        }
    }
    return H;
}



int main()
{
//double lamb = 3.1;
double h[10] = {0.01,0.02,0.04,0.08,0.3,1.2,5.0,7.0,10.0,15.0};
double J = 1.0;
double lamb[17] = {2.0,2.2,2.3,2.4,2.6,2.7,2.8,3.0,3.2,3.3,3.4,3.5,3.6,3.7,3.8,3.9,4.0};
for(int b = 0; b < 10; b++){
        for(int c = 0; c < 17; c++){
ofstream myfile;
myfile.open ("l=101,j=1,lambda=" + to_string(lamb[c]) + ",h=" + to_string(h[b]) + ".csv")
    ↪ ;
long double data[72][3];
cout << "radius" << ";␣" << "Circle␣positive␣component" << ";␣" << "Lambda-circle␣
    ↪ positive␣component" << ";␣" << "Square␣positive␣component" << "\n␣";
myfile << "radius" << ";␣" << "Circle␣positive␣component" << ";␣" << "Lambda-circle␣
    ↪ positive␣component" << ";␣" << "Square␣positive␣component" << "\n␣";
for(int a = 0; a < 72; a++){
    cout << a << ";␣";
    myfile << a << ";␣";
    data[a][0] = Hamiltonian(statewithcirclecomponent(50,a),J,lamb[c],h[b]);
    cout << data[a][0] << ";␣";
    myfile << data[a][0] << ";␣";
    data[a][1] = Hamiltonian(statewithlambdacomponent(50,a,lamb[c]),J,lamb[c],h[b]);
    cout << data[a][1] << ";␣";
    myfile << data[a][1] << ";␣";
    data[a][2] = Hamiltonian(statewithsquarecomponent(50,a),J,lamb[c],h[b]);
    cout << data[a][2] << "\n␣";
    myfile << data[a][2] << "\n␣";
}

myfile.close();
        }
}
//Beep(523,500); // 523 hertz (C5) for 500 milliseconds

return 0;
}
```

B Algorithm for finding the gateway shape

```
 #include <iostream>
#include <cmath>
#include <fstream>
#include <math.h>
#include <windows.h>
#include <vector>
#include <array>
#include <algorithm>
#include <bits/stdc++.h>
#include <random>
#include <chrono>
#include <tuple>

using namespace std;
auto seed = chrono::high_resolution_clock::now().time_since_epoch().count();
mt19937 mt_rand(seed);
auto real_rand = std::bind(std::uniform_real_distribution<double>(0,1),mt19937(seed));
auto int_rand = std::bind(std::uniform_int_distribution<int>(0,100000),mt19937(seed));
int maximumsizeofcomponent = 1400;
int numberofrepetitions = 200000;
double h = 0.6;
double J = 1.0;
double lambda = 2.9;
int length = 101;

int mod(int x, int m);
vector<double> Update_Hamiltonian(vector<pair<int,int> > positive_component, vector<
    ↪ double> Hamiltonian, pair<int, int> new_vertice);
pair<pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > >, vector<double> >
    ↪ component_forming_algorithm(pair<pair<vector<pair<int,int> >, vector<pair<pair<int
    ↪ ,int>,int> > >, vector<double> > information);
double one_spin_H();

int main()
{
double onespin=one_spin_H();
cout << onespin;
vector< double> H {0.0};
vector< pair< pair<int, int>, int> > border_set;
vector< pair<int, int > > component;
component.push_back({length/2,length/2});
border_set.push_back ({{length/2+1,length/2},1});
border_set.push_back ({{length/2-1,length/2},1});
border_set.push_back ({{length/2,length/2+1},1});
border_set.push_back ({{length/2,length/2-1},1});
pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > >
    ↪ starting_lattice_information = {component,border_set};
pair<pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > >, vector<double> >
    ↪ starting_total_information = {starting_lattice_information,H};
pair<pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > >, vector<double> >
    ↪ end_result;
vector<pair<vector<pair<int,int> >,vector<double> > > list_of_simulations;
int number_of_simulations = numberofrepetitions;
for(int a_it = 0; a_it < number_of_simulations; a_it++){
end_result = component_forming_algorithm(starting_total_information);
vector<pair<int,int> > end_component = (end_result.first).first;
vector<double> Hamiltonian = end_result.second;
pair<vector<pair<int,int> >,vector<double> > result_of_simulation {end_component,
    ↪ Hamiltonian};
list_of_simulations.push_back(result_of_simulation);
cout << a_it << "\n";
}
end_result = component_forming_algorithm(starting_total_information);
vector<pair<int,int> > end_component = (end_result.first).first;
vector<double> Hamiltonian = end_result.second;

ofstream myfile;
myfile.open ("resultofsimulations,␣#␣=␣" + to_string(number_of_simulations)+ ",lamb=" +
    ↪ to_string(lambda) + ",h=" + to_string(h) + ",length="+to_string(length)+",
    ↪ maxsizeofcomponent="+ to_string(maximumsizeofcomponent)+".csv");
for(int a = 0; a < number_of_simulations; a++){
        for(int b = 0; b < maximumsizeofcomponent; b++) {
            myfile << "(" << ((list_of_simulations[a].first)[b]).first << "," << ((
                ↪ list_of_simulations[a].first)[b]).second << "," << (list_of_simulations
                ↪ [a].second)[b] << ");";
    }
    cout << a << "\n";
    myfile << "\n";
}
myfile.close();

return 0;
}
double one_spin_H(){
double onespin=0;
if (length%2 == 1) {
    for (int a = 1; a < length/2+1 ; a++){
        for(int b = 0; b < length/2+1; b++ ){
            onespin = onespin + 4/(pow(pow(a,2)+pow(b,2),(lambda/2)));
        }
    }
    return onespin;
} else {
    for (int a = 1; a < length/2 ; a++){
        for(int b = 0; b < length/2; b++ ){
            onespin = onespin + 4/(pow(pow(a,2)+pow(b,2),(lambda/2)));
        }
        onespin = onespin + 4/(pow(pow(a,2)+pow(length/2,2),(lambda/2)));
    }
    onespin = onespin + 2/(pow(pow(length/2,2),(lambda/2)));
    onespin = onespin + 1/(pow(pow(length/2,2)+pow(length/2,2),(lambda/2)));
    return onespin;
}
}
```

```
int mod(int x, int m) {
    return (x%m + m)%m;
}

vector<double> Update_Hamiltonian(vector<pair<int,int> > positive_component, vector<
    ↪ double> Hamiltonian, pair<int, int> new_vertice) {
double onespin = one_spin_H();
double H = Hamiltonian.back() - 2*h + 2*J*onespin;
for (int a = 0; a < positive_component.size()-1; a++){
        H = H -(4*J)/(pow(pow(min(abs(positive_component[a].first-new_vertice.first), min(
            ↪ abs(positive_component[a].first-new_vertice.first+length), abs(
            ↪ positive_component[a].first-new_vertice.first-length))),2)+ pow(min(abs(
            ↪ positive_component[a].second-new_vertice.second), min(abs(
            ↪ positive_component[a].second-new_vertice.second+length), abs(
            ↪ positive_component[a].second-new_vertice.second-length))),2),(lambda/2)));
}
Hamiltonian.push_back(H);
return Hamiltonian;
}

vector<pair<pair<int,int>,int> > Update_Border(pair<int,int> vertex, vector<pair<int,int>
    ↪  > component, vector<pair<pair<int,int>, int> > border_set){
        auto it_vertex = std::find_if (
            border_set.begin(),
            border_set.end(),
            [&vertex](pair< pair<int, int>, int> border_vertice) -> bool
                                {
                                    return border_vertice.first == vertex;
                                    }
        );

        border_set.erase(it_vertex);
        //
        // FIRST BORDER LOCATION
        //
        pair<int, int> first_border {mod(vertex.first+1,length), vertex.second};
        auto component_check_1 = std::find (
            component.begin(),
            component.end(),
            first_border
        );
        if(component_check_1 == component.end()){
        auto it_1 = std::find_if (
            border_set.begin(),
            border_set.end(),
            [&first_border](pair< pair<int, int>, int> border_vertice) -> bool
                                {
                                    return border_vertice.first == first_border;
                                    }
        );

        if (it_1 != border_set.end()) {
            (*it_1).second = (*it_1).second + 1;
        } else{
```

```
    border_set.push_back({first_border,1});
}
}

// SECOND BORDER LOCATION

pair<int, int> second_border {mod(vertex.first-1,length), vertex.second};
auto component_check_2 = std::find (
    component.begin(),
    component.end(),
    second_border
);
if(component_check_2 == component.end()){
auto it_2 = std::find_if (
    border_set.begin(),
    border_set.end(),
    [&second_border](pair< pair<int, int>, int> border_vertice) -> bool
                        {
                            return border_vertice.first == second_border;
                            }
);

if (it_2 != border_set.end()) {
    (*it_2).second = (*it_2).second + 1;
} else{
    border_set.push_back({second_border,1});
}
}

// THIRD BORDER LOCATION

pair<int, int> third_border {vertex.first, mod(vertex.second+1,length)};
auto component_check_3 = std::find (
    component.begin(),
    component.end(),
    third_border
);
if(component_check_3 == component.end()){
auto it_3 = std::find_if (
    border_set.begin(),
    border_set.end(),
    [&third_border](pair< pair<int, int>, int> border_vertice) -> bool
                        {
                            return border_vertice.first == third_border;
                            }
);

if (it_3 != border_set.end()) {
    (*it_3).second = (*it_3).second + 1;
} else{
    border_set.push_back({third_border,1});
}
}
```

```
        // FOURTH BORDER LOCATION
        pair<int, int> fourth_border {vertex.first, mod(vertex.second-1,length)};
        auto component_check_4 = std::find (
            component.begin(),
            component.end(),
            fourth_border
        );
        if(component_check_4 == component.end()){
        auto it_4 = std::find_if (
            border_set.begin(),
            border_set.end(),
            [&fourth_border](pair< pair<int, int>, int> border_vertice) -> bool
                              {
                                  return border_vertice.first == fourth_border;
                                  }
        );

        if (it_4 != border_set.end()) {
            (*it_4).second = (*it_4).second + 1;
        } else{
            border_set.push_back({fourth_border,1});
        }
        }

        return border_set;
}

pair<pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > >, vector<double> >
    ↪ component_forming_algorithm(pair<pair<vector<pair<int,int> >, vector<pair<pair<int
    ↪ ,int>,int> > >, vector<double> > information){
    int sizeofcomponent = maximumsizeofcomponent;
    vector<pair<int,int> > component = (information.first).first;
    vector<pair<pair<int,int>,int> > border_set = (information.first).second;
    vector<double> H = information.second;

    if (component.size() == sizeofcomponent) {
        return information;
    } else{

        bool added_a_three = false;

        for(int a = 0; a < border_set.size(); a = a + 1 ) {
            if(border_set[a].second == 3){
                pair<int, int> vertex = border_set[a].first;
                component.push_back(vertex);
                added_a_three = true;

                H = Update_Hamiltonian(component, H, vertex);
                border_set = Update_Border(vertex, component, border_set);

            break;

            }
        }
    if (added_a_three == false){
        double randomNumber = real_rand();
        auto it_border1 = std::find_if(std::begin(border_set), std::end(border_set),
            ↪ [](pair<pair<int,int>, int> border_element){return border_element.
            ↪ second == 1;});
        auto it_border2 = std::find_if(std::begin(border_set), std::end(border_set),
            ↪ [](pair<pair<int,int>, int> border_element){return border_element.
            ↪ second == 2;});
        if(it_border1== std::end(border_set)){
        randomNumber=0.0;
        } else if (it_border2==std::end(border_set)){
        randomNumber=1.0;
        }
        bool whichonewechoose = randomNumber < 0.66;
        if (randomNumber < 0.66) {
            std::vector<size_t> results;
            auto it = std::find_if(std::begin(border_set), std::end(border_set), [](
                ↪ pair<pair<int,int>, int> border_element){return border_element.
                ↪ second == 2;});
            while (it != std::end(border_set)) {
            results.emplace_back(std::distance(std::begin(border_set), it));
            it = std::find_if(std::next(it), std::end(border_set), [](pair<pair<int,int
                ↪ >, int> border_element){return border_element.second == 2;});
            }
            int random_vertice_location = mod(int_rand(),results.size());
            pair<int,int> selected_border_element = border_set[results[
                ↪ random_vertice_location]].first;

            component.push_back(selected_border_element);
            H = Update_Hamiltonian(component, H, selected_border_element);
            border_set = Update_Border(selected_border_element, component,border_set);
        } else{
            std::vector<size_t> results;
            auto it = std::find_if(std::begin(border_set), std::end(border_set), [](
                ↪ pair<pair<int,int>, int> border_element){return border_element.
                ↪ second == 1;});
            while (it != std::end(border_set)) {
            results.emplace_back(std::distance(std::begin(border_set), it));
            it = std::find_if(std::next(it), std::end(border_set), [](pair<pair<int,int
                ↪ >, int> border_element){return border_element.second == 1;});
            }
            int random_vertice_location = mod(int_rand(),results.size());
            pair<int,int> selected_border_element = border_set[results[
                ↪ random_vertice_location]].first;

            component.push_back(selected_border_element);
            H = Update_Hamiltonian(component, H, selected_border_element);
            border_set = Update_Border(selected_border_element, component,border_set);
        }

    }
pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > > vertex_information =
    ↪ {component,border_set};
    pair<pair<vector<pair<int,int> >, vector<pair<pair<int,int>,int> > >, vector<double>
        ↪ > new_information = {vertex_information,H};
    return component_forming_algorithm(new_information);
    }
}
```

C Metropolis-Glauber algorithm

```
 #include <iostream>
#include <cmath>
#include <fstream>
#include <math.h>
#include <windows.h>
#include <vector>
#include <array>
#include <algorithm>
#include <bits/stdc++.h>
#include <random>
#include <chrono>
#include <tuple>

using namespace std;
auto seed = chrono::high_resolution_clock::now().time_since_epoch().count();
mt19937 mt_rand(seed);
auto real_rand = std::bind(std::uniform_real_distribution<double>(0,1),mt19937(seed));
auto int_rand = std::bind(std::uniform_int_distribution<int>(0,100000),mt19937(seed));
double h = 2.0;
double J = 1.0;
double lambda = 2.9;
int length = 100;
double beta = 0.27;
long long int tau=0;

double one_spin_H(){
double onespin=0;
if (length%2 == 1) {
    for (int a = 1; a < length/2+1 ; a++){
        for(int b = 0; b < length/2+1; b++ ){
            onespin = onespin + 4/(pow(pow(a,2)+pow(b,2),(lambda/2)));
        }
    }
    return onespin;
} else {
    for (int a = 1; a < length/2 ; a++){
        for(int b = 0; b < length/2; b++ ){
            onespin = onespin + 4/(pow(pow(a,2)+pow(b,2),(lambda/2)));
        }
        onespin = onespin + 4/(pow(pow(a,2)+pow(length/2,2),(lambda/2)));
    }
    onespin = onespin + 2/(pow(pow(length/2,2),(lambda/2)));
    onespin = onespin + 1/(pow(pow(length/2,2)+pow(length/2,2),(lambda/2)));
    return onespin;
}
}

double onespin = one_spin_H();

double Delta_Hamiltonian(vector<pair<int,int> > component, pair<int,int> new_vertice) {
    vector<pair<int,int> > temp_component = component;
    /*double onespin = one_spin_H();*/
    auto it = std::find(begin(temp_component),end(temp_component), new_vertice);
    int sigma = 1;
    if (it == temp_component.end()){
        sigma = -1;
     } else {
        sigma = 1;
        temp_component.erase(it);
    }
    double H = - 2*h + 2*J*onespin;
    for (int a = 0; a < temp_component.size(); a++){
        double nomer = pow(pow(min(abs(temp_component[a].first-new_vertice.first), min(abs
            ↪ (temp_component[a].first-new_vertice.first+length), abs(temp_component[a].
            ↪ first-new_vertice.first-length))),2)+ pow(min(abs(temp_component[a].second-
            ↪ new_vertice.second), min(abs(temp_component[a].second-new_vertice.second+
            ↪ length), abs(temp_component[a].second-new_vertice.second-length))),2),(
            ↪ lambda/2));
        if (nomer == 0){
            cout << "ERROR␣DIVIDE␣BY␣ZERO!";
        }
        H = H -(4*J)/nomer;
    }
    H = -(sigma * H);
    return H;
}

int main()
{
vector<int> componentsize = {};
pair<int, int> starting_point = {length/2,length/2};
vector<pair<int,int> > component = {};
component.push_back(starting_point);
int maxsize = 0;
while (component.size() < (0.9*length*length)) {
        if (tau%100000==0) {
            cout<< "tau␣=␣" << tau << ",␣component␣size␣=␣" << component.size() << ",␣
                ↪ maximum␣component␣size␣=␣" << maxsize << "\n";
        }
        int x_component = int_rand()%length;
        int y_component = int_rand()%length;
        pair<int,int> new_vertice = {x_component,y_component};

        auto it = std::find(begin(component), end(component), new_vertice);

        double delt_ham = Delta_Hamiltonian(component, new_vertice);
        if (delt_ham < 0) {
            if (it == component.end()){
                /*cout << "delt_ham < 0 ,new vertice was not in component, new element
                    ↪ added component size = " << static_cast<int>(component.size()) << ",
                    ↪  tau = " << tau ;*/
                component.push_back(new_vertice);
                componentsize.push_back(component.size());
                if (component.size() > maxsize) {
                    maxsize = component.size();
                }
                /*cout << ", maximumum size = " << maxsize << "\n";*/
            } else{
                component.erase(it);
                /*cout << "delt_ham < 0, new vertice was in component, element removed
                    ↪ component size = "<< static_cast<int>(component.size())<< ", tau = "
                    ↪  << tau ;*/
                componentsize.push_back(component.size());
                    if (component.size() > maxsize) {
                    maxsize = component.size();
                }
                /*cout << ", maximumum size = " << maxsize << "\n";*/
            }
        } else {
            double threshold = exp((-beta)*delt_ham);
            double rv = real_rand();
            if (rv < threshold){
                if (it == component.end()){
                    /*cout << "delt_ham >= 0, new vertice was not in component, vertice
                        ↪ added component size = " << static_cast<int>(component.size())
                        ↪ << ", tau = " << tau ;*/
                    component.push_back(new_vertice);
                    componentsize.push_back(component.size());
                    if (component.size() > maxsize) {
                    maxsize = component.size();
                }
                /*cout << ", maximumum size = " << maxsize << "\n";*/
                } else{
                    component.erase(it);
                    /*cout << "delt_ham >= 0, new vertice was in component, vertice removed
                        ↪ , component size = " << static_cast<int>(component.size()) << ",
                        ↪  tau = " << tau ;*/
                    componentsize.push_back(component.size());
                    if (component.size() > maxsize) {
                    maxsize = component.size();
                }
                /*cout << ", maximumum size = " << maxsize << "\n";*/
                }
            }
        }
        tau = tau+1;

    }



cout << "This␣is␣the␣end␣time:␣";
cout << tau;

ofstream myfile;
myfile.open("glauber_simulation,␣length␣=␣␣"+ to_string(length)+ ",␣beta␣=␣"+ to_string(
    ↪ beta) +",␣h=" + to_string(h) + ",␣lambda␣=␣" + to_string(lambda) + ".csv");
int filesize = componentsize.size();
myfile << tau << "\n";
for (int a = 0; a < filesize; a++) {
    myfile << componentsize[a] << ";";
}


return 0;
}
```

References

[1] David J Aldous. Some inequalities for reversible markov chains. *Journal of the London Mathematical Society*, 2(3):564–576, 1982.

[2] S. Blundell, S.J. Blundell, and K.M. Blundell. *Concepts in Thermal Physics*. Oxford University Press, 2006.

[3] Anton Bovier and Frank den Hollander. Metastability: a potential theoretic approach. In *International Congress of Mathematicians*, volume 3, pages 499–518. Eur. Math. Soc. Zürich, 2006.

[4] Emilio NM Cirillo and Joel L Lebowitz. Metastability in the two-dimensional ising model with free boundary conditions. *Journal of Statistical Physics*, 90(1-2):211–226, 1998.

[5] Francesco Manzo, Francesca R Nardi, Enzo Olivieri, and Elisabetta Scoppola. On the essential features of metastability: tunnelling time and critical configurations. *Journal of Statistical Physics*, 115(1-2):591–642, 2004.

[6] Antonietta Mira and Charles J Geyer. On non-reversible markov chains. *Monte Carlo Methods, Fields Institute/AMS*, pages 95–110, 2000.

[7] Dae-Yup Song. Tunneling and energy splitting in an asymmetric double-well potential. *Annals of Physics*, 323(12):2991–2999, 2008.

[8] Aernout C.D. van Enter, Bruno Kimura, Wioletta Ruszel, and Cristian Spitoni. Nucleation for one-dimensional long-range ising models. *Journal of Statistical Physics*, 174(6):1327–1345, 2019.