Investigation of the radius of convergence for the two dimensional Ising model

Alexander Adam

BERGISCHE
UNIVERSITÄT
WUPPERTAL

Bachelor thesis submitted in partial fulfilment of the requirements to achieve the degree of Bachelor of Science

---

Theoretical particle physics

Faculty of Mathematics and Natural Sciences

Bergische Universität Wuppertal

First supervisor:
Jun. Prof. Dr. J. Günther

Second supervisor:
Prof. Dr. S. Borsányi

Hand in Date:
April 17, 2023

Contents

List of Figures

List of Tables

Listings

Acknowledgments

First and foremost I would like thank Prof. Dr. Jana N. Günther and Prof. Dr. Szabolcs Borsányi for the support as well as the challenging and exciting thesis topic. Furthermore, I would like to thank Nuha, Fabian and Ali for the professional discussions and support. The conversations at the weekly group meetings have also been valuable.

1 Introduction

The theory of the strong interaction, quantum chromodynamics (QCD), is expected to exhibit a second-order phase transition. However, due to the limitations of current computational resources, it is not possible to directly simulate the region where the critical point, i.e. the second-order phase transition, is expected to occur, making it difficult to determine its precise location. To overcome this challenge, methods have been developed to estimate the location of the critical point from simulations carried out further from the critical point. One such method is to analyse the radius of convergence of applicable observables and use their divergence at the critical point to determine its position [1]. Of course simulations in quantum chromodynamics as well as the method itself are complicated and computationally intensive.

We want to take a step back and apply the method to a model that is both easier to simulate and easier to understand and also allows comparisons with known values along the way. For this purpose, we use the Ising model, a mathematical model that can be used to describe ferromagnetism. Specifically, we are interested in the two-dimensional case. It was first written down in 1920 by W. Lenz, with Onsager formulating a solution in 1944 for the two-dimensional case without an external magnetic field [2]. The rationale for using the two-dimensional Ising model is that it is a simple yet powerful model for understanding critical phase transitions. In addition it is easy to simulate, analytically solvable and includes a second order phase transition. Therefore, the two-dimensional Ising model is ideal to test the method for determining the critical point using the radius of convergence, as it allows the simulated results to be compared with the corresponding theoretical values. The aim of this thesis is therefore to estimate the critical inverse temperature β_c at which the critical point is located, using Monte Carlo simulations from which the radius of convergence can be calculated. More precisely, we will use the Fisher zeros, whose estimation will rely on the radii of convergence in conjunction with there finite size scaling, to determine the critical exponent ν, a parameter dictating the aformentiod scaling, and the critical inverse temperate β_c of the two-dimensional Ising model.

We will first describe the pure mathematics used to find two expressions for calculating the convergence radius based on a Taylor series. Subsequently, the physical basis used to describe models such as the Ising model, i.e. statistical mechanics, are explained. This chapter will also include the application of statistical mechanics to the Ising model and the resulting expressions, such as the partition sum or the theoretical value of the critical inverse temperature. In addition, an overview of phase transitions and Fisher zeros is given, as well as an explanation of the method used in this thesis and how it can be applied to determine the critical exponent ν and the critical inverse temperature β_c. This is followed by an explanation of the main numerical algorithms used in this thesis, together with their implementation in the Julia programming language. The implementation of the Wolff algorithm used to simulate the Ising model is explained along with various optimisations. In addition, an error in the code written using the Wolff algorithm, which causes deviations from theory for small lattice sizes, is reversed. Furthermore, the method for estimating the statistical error is explained and the code for calculating the derivative for Taylor series is presented. Lastly, the results of the

simulations are interpreted using the knowledge gained from the chapters about the statistical mechanics. It starts with a verification for small and large lattice sizes, using the exact partition function and Onsager's solution as values to compare. Following the computed radii of convergence, Fisher zeros, critical exponent ν and critical inverse temperature β_c are presented and compared to known values. It ends with a determination of the total error of ν and β_c using a method from [3]. To summarise the objective, we want to conclude whether or not the method described, using the radii of convergence, can be used to determine the location of the critical point in the Ising model.

2 Mathematical Foundation

This chapter summarizes the elementary, yet important mathematical concepts in a concise way . It is partly based on [4].

Let f be a function that is infinitely differentiable on the real interval (a, b) and x_0 be a point inside the aforementioned interval. One can then define the so called Taylor polynomial as follows

$$T_N f(x; x_0) := \sum_{n=0}^{N} \frac{f^{(n)}(x_0)}{n!}(x - x_0)^n. \tag{2.1}$$

As N goes to infinity, the function f can be expressed, at the expansion point x_0, as a power series.

$$f(x; x_0) = \sum_{n=0}^{\infty} c_n(x - x_0)^n \qquad c_n = \frac{f^{(n)}(x_0)}{n!}. \tag{2.2}$$

This power series is called the Taylor series of f at point x_0. While it is certain that a Taylor series converges to the exact value at the expansion point x_0, whether the series converges for a point different from the expansion point, depends on the function f. To check if the Taylor series converges, the root test can be used. It states that a sum of coefficients a_n converges if the value of C, as defined as $\limsup_{n\to\infty} \sqrt[n]{|a_n|}$, is less than 1 and diverges if it is greater than 1. If it is equal to 1, the test is inconclusive. Applying this test to the Taylor series yields:

$$\begin{aligned} C = \limsup_{n\to\infty} \sqrt[n]{|a_n|} &= \limsup_{n\to\infty} \sqrt[n]{|c_n(x - x_0)^n|} \\ &= \limsup_{n\to\infty} \left(\sqrt[n]{|c_n|} \right) |x - x_0|. \end{aligned}$$

Using the constraint $C < 1$ for the convergence and the definition of the Taylor series, the maximum value of $|x - x_0|$, henceforth called the radius of convergence r, is given by

$$r = \frac{1}{\limsup_{n\to\infty} \left(\sqrt[n]{\left| \frac{f^{(n)}(x_0)}{n!} \right|} \right)}. \tag{2.3}$$

This equation is the essence of the Cauchy–Hadamard theorem. Under the assumption that each of the Taylor coefficients are non zero, the equation can be rewritten as follows

$$r = \liminf_{n\to\infty} \left(\sqrt[n]{\left| \frac{n!}{f^{(n)}(x_0)} \right|} \right). \tag{2.4}$$

Another method to determine the radius of convergence is to use the ratio test. The ratio test states that a sum of coefficients a_n converges if

$$\lim_{n\to\infty} \left| \frac{a_n}{a_{n+1}} \right| < 1. \tag{2.5}$$

Carrying out the ratio test for a Taylor series lead to:

$$1 > \lim_{n\to\infty}\left|\frac{a_{n+1}}{a_n}\right| = \lim_{n\to\infty}\left|\frac{c_{n+1}(x-x_0)^{(n+1)}}{c_n(x-x_0)^n}\right| = \lim_{n\to\infty}\left|\frac{c_{n+1}}{c_n}\right||x-x_0|.$$

As with the root test, the ratio test gives an upper bound on $|x - x_0|$ in the limit of $n \to \infty$. The radius of convergence r derived from the ratio test is

$$r = \lim_{n\to\infty}\left|\frac{c_n}{c_{n+1}}\right|. \tag{2.6}$$

One can show that the radius of convergence is equal to the infimum of the distance between the expansion point x_0 and the singularities of the function f.

It is important to mention that even if one only looks at the real axis, singularities inside the complex plane still affect the radius of convergence. As an example, consider

$$f(z) = \frac{1}{1+z^2} = \frac{1}{2}\Big(\frac{1}{z+i} + \frac{1}{z-i}\Big). \tag{2.7}$$

It is continuous along the whole of the real axis, while at the same time having two poles at i and $-i$ in the complex plane. The n^{th} derivative at $z = 0$ can be computed to be

$$\left[\frac{\partial^n f}{\partial z^n}\right]_{z=0} = \frac{-in!}{2}\Big((-i)^{n-1} - i^{n-1}\Big) \tag{2.8}$$

from which the Taylor series at $z = 0$ follows:

$$\sum_{n=0}^{\infty}\frac{1}{2}((-i)^n - i^n)(z-0)^n. \tag{2.9}$$

Applying the root test to calculate the convergence radius leads to:

$$r = \frac{1}{\limsup_{n\to\infty}\left(\sqrt[n]{\left|\frac{f^{(n)}(x_0)}{n!}\right|}\right)} = \frac{1}{\limsup_{n\to\infty}\left(\sqrt[n]{0.5((-i)^n - i^n)}\right)}$$

Using the fact that $(i)^n$ & $(-i)^n$ are bounded as well as the fact that the limit of $\sqrt[n]{C}$, where C is a constant converges to 1, the lim sup can be determined to be 1. This leads to a convergence radius of 1 which, as expected, is equal to the distance to the poles at i & $-i$. It shows that functions continuous along the entire real axis can still have a finite radius of convergence for a given point on the real axis.

3 Statistical Mechanics

Statistical mechanics studies systems whose many degrees of freedom make it impossible to use the equations of motion to analyse and understand their dynamics. Instead, statistical mechanics uses statistical ensembles to describe all possible states of a system. These ensembles themselves only depend on a manageable number of observables, such as temperature, internal energy, or particle count. Depending on the constraints placed on a system, different statistical ensembles can be used to describe the macroscopic properties of the given system. Due to the fact that this thesis is only concerned with the Ising model in which just the energy can vary, only the canonical ensemble is investigated in more detail. It should be noted, nevertheless, that in the thermodynamic limit all ensembles lead to the same results.

3.1 Canonical Ensemble

In the canonical ensemble, as stated before, only the energy varies while all other parameters stay constant, such as the volume, temperature or particle count. The partition function of the canonical ensemble is defined as

$$Z = \sum_i e^{-\beta E_i}, \tag{3.1}$$

where the sum over i implies a summation over all possible states of the system. In addition, E_i is the energy of the respective state i and β is defined as the inverse temperature $1/(kT)$, where T is the absolute temperature and k is the Boltzmann constant. Any expectation value of an observable O can be measured by averaging its measurements over all possible states with the weight $\exp(-\beta E_i)$, also called the Boltzmann weight. The weighting ensures that unlikely states do not skew the expected value.

$$\langle O \rangle = \frac{\sum_i O_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}} = \frac{1}{Z} \sum_i O_i e^{-\beta E_i} \tag{3.2}$$

For the energy E of the system, Equation 3.2 can be expressed in a more compact way by using the fact that the observable itself appears in the weight. It follows that the expected value of the energy is defined as follows in Equation 3.3.

$$\langle E \rangle = \frac{1}{Z} \sum_i E_i e^{-\beta E_i} = \frac{1}{Z}(-\partial_\beta) \sum_i e^{-\beta E_i} = -\frac{\partial}{\partial \beta} \log(Z(\beta)) \tag{3.3}$$

The equation above enables one to calculate the exact expectation value of the energy from the partition function, assuming the partition function can be written in a concise way and the derivative w.r.t. the inverse temperature can be carried out. The internal energy e is defined as E/V.

3.1.1 Recursive differentiation

To calculate the derivatives of observables such as the internal energy of systems where the partition function is not known or cannot be easily differentiated, Equation 3.2 can be used to construct a general formula for the derivative.

$$\begin{aligned}
\frac{\partial}{\partial\beta}\langle O\rangle &= \frac{\partial}{\partial\beta}\frac{\sum_i O_i e^{-\beta E_i}}{\sum_i e^{-\beta E_i}} \\
&= \frac{\left(\sum_i O_i(-E_i)e^{-\beta E_i}\right)\left(\sum_i e^{-\beta E_i}\right) - \left(\sum_i O_i e^{-\beta E_i}\right)\left(\sum_i(-E_i)e^{-\beta E_i}\right)}{(\sum_i e^{-\beta E_i})(\sum_i e^{-\beta E_i})} \\
&= \frac{\left(\sum_i(-E_i)O_i e^{-\beta E_i}\right)}{(\sum_i e^{-\beta E_i})} + \frac{\left(\sum_i O_i e^{-\beta E_i}\right)}{(\sum_i e^{-\beta E_i})}\frac{\left(\sum_i E_i e^{-\beta E_i}\right)}{(\sum_i e^{-\beta E_i})} \\
&= -\langle EO\rangle + \langle O\rangle\langle E\rangle
\end{aligned}$$

As one can see, taking the derivative of Equation 3.2 leads to an expression which itself contains only the expectation values of O,E and their product. The consequence is that for the calculation of higher order derivatives one simply has to reapply the definition of the derivative. The derivation of the formula for the second order derivative can be seen in section A.1 along with the expressions for the first six derivatives of the internal energy which were calculated symbolically.

3.2 The two-dimensional Ising model

The Ising model is a widely used mathematical model that provides valuable insights into the behavior of physical systems. This simple yet effective model is based on a lattice, with each point on the lattice representing a spin that can be oriented either up or down. To be able to write down equations such as the Hamiltonian or the partition function, we define Λ to be the set of all lattice sites. For each lattice site $i \in \Lambda$, σ_i indicates whether the spin of the site is up $(+1)$ or down (-1). A whole collection of these spins is written as σ.

The Ising model is often used to simulate ferromagnetic materials, which are materials that become magnetized in the presence of an external magnetic field. In two dimensions, the Hamiltonian of the Ising model is determined by two key factors: the strength of the interaction between neighboring spins (represented by the coefficient J_{ij}) and the coupling coefficient (represented by h_j) that describe the interaction between the spin and an external magnetic field. Based on this description, the Hamiltonian can then be written as seen in Equation 3.4.

$$H_{general} = -\sum_{i,j\in\Lambda} J_{ij}\sigma_i\sigma_j - \sum_{j\in\Lambda} h_j\sigma_j \tag{3.4}$$

In the following, however, we will only consider nearest neighbour interactions with a constant interaction strength and a vanishing external magnetic field. The general Hamiltonian from Equation 3.4 is simplified under the above constraints to Equation 3.5,where $<ij>$ indicates that sites i and j are nearest neighbours.

$$H_{simple} = -J\sum_{<ij>}\sigma_i\sigma_j \tag{3.5}$$

Assuming periodic boundary conditions and a lattice of size L^2, the partition function can then be described as follows, where $\{\sigma\}$ represents the set of all possible spin configurations

$$Z(\beta, J) = \sum_{\{\sigma\}} e^{-\beta H(\sigma)} = \sum_{\{\sigma\}} \exp\left\{ \beta J \sum_{x,y \in L} (\sigma_{x,y}(\sigma_{x,y+1} + \sigma_{x+1,y})) \right\}. \tag{3.6}$$

The simplifications are made because the Ising model can now be solved analytically in two dimensions in the thermodynamic limit so that the simulated results can later be compared. Analytical solutions for finite but arbitrary lattices have been formulated, e.g. in [5], but they require free boundary conditions and are therefore not applicable. It is important to note that the simplified two-dimensional system still has a phase transition, unlike the one-dimensional case. The analytical solution was first written down by Onsager[6] in terms of free energy. Equation 3.7 shows his solution where J_1 and J_2 have already been set to J, i.e. the isotropic case.

$$\begin{aligned} -\beta f_{ons} = \ln 2 + \frac{1}{8\pi^2} \int_0^{2\pi} d\theta_1 \int_0^{2\pi} d\theta_2 \ln[\cosh(2\beta J)^2 \\ - \sinh(2\beta J)(\cos(\theta_1) + \cos(\theta_2))] \end{aligned} \tag{3.7}$$

Another equivalent formulation of Onsager's using only a single integral is as follows

$$-\beta F = \frac{\ln 2}{2} + \frac{1}{2\pi} \int_0^{\pi} d\theta \ln[\cosh(2\beta J_1)\cosh(2\beta J_2) + \frac{1}{k}\sqrt{1 + k^2 - 2k\cos(2\theta)}] \tag{3.8}$$

with k being defined as $\sinh(2\beta J)^{-2}$. Based on this equation of the free energy, Onsager calculated the internal energy per site to be

$$e = -J\coth(2\beta J)\left[1 + \frac{1}{\pi}(2\tanh(2\beta J)^2 - 1)\int_0^{\pi/2} \frac{1}{\sqrt{(1 - 4k(1+k)^{(-2)}\sin(\theta)^2}}\right]. \tag{3.9}$$

In addition to the free and internal energy, Onsager was also able to obtain the critical inverse temperature [1]

$$\beta_c = \frac{\ln\left(1 + \sqrt{2}\right)}{2J}. \tag{3.10}$$

For $J = 1$ one obtains $\beta_c \approx 0.4407$. While Onsager's solution is correct for the case of a infinite volume, it is unable to predict the results for finite lattices due to artifacts arising form the finite volume, so called finite volume errors. For larger lattices this deviation is negligible, but for smaller grids in the order of 10 or 20 (depending on the temperature, at which the simulation is performed) the deviation is significant and does not allow a comparison of the simulation with the analytic solution. However, if we look at lattice sizes in the low single digits, we can calculate the partition function directly and thus compare the results. As an example, the calculation of the partition

[1] The critical (inverse) temperature can be determined without needing the full analytical solution of the Ising model, eg see [7]

sum for a lattice of size 2^2 follows. While for lattice sizes of 2^2 and 4^2 the partition function were calculated symbolically, the calculations for a lattice of size 6^2 are based on [8]. The external source was used to obtain it, as it becomes unfeasible to calculate the partition function via a simple brute force algorithm. [8] was also used to verify the correctness of the calculated partition function for the sizes 2^2 and 4^2.

For a lattice size of 2^2 there are 16 ($= 2^{(2^2)}$) possible spin configurations, since on the 2^2 lattice each point can take on two values (spin up or spin down). Below are the spin configurations listed with an energy that differs from zero:

$$H = -8J \quad \begin{vmatrix} \downarrow & \downarrow \\ \downarrow & \downarrow \end{vmatrix} \qquad H = 8J \quad \begin{vmatrix} \downarrow & \uparrow \\ \uparrow & \downarrow \end{vmatrix} \qquad H = 8J \quad \begin{vmatrix} \uparrow & \downarrow \\ \downarrow & \uparrow \end{vmatrix} \qquad H = -8J \quad \begin{vmatrix} \uparrow & \uparrow \\ \uparrow & \uparrow \end{vmatrix}$$

All the remaining 12 configurations have an energy that equals zero. The resulting partition function can be seen in Equation 3.11

$$Z_{L=2} = 2e^{\beta J 8} + 12 + 2e^{-\beta J 8} \tag{3.11}$$

3.2.1 Phase transition and Fisher zeros

This subsection loosely describes what phase transitions are in statistical mechanics. Definitions introduced here may not be complete and will only serve to give a rough overview, while leaving out some details.

Generally speaking, systems which have multiple phases, can exhibit so called first or second order phase transitions when changing from one phase to another. To quantify the concept of phases, one introduces so called order parameters, which are observables that help to distinguish between the phases, e.g. the derivative of the free energy with respect to the external field. For the Ising model this would be the magnetization of the system, defined as the sum of all spins. Using the Ehrenfest classification, a first-order phase transition is defined as those lines in the phase diagram, where the first derivative of the free energy is discontinuous. Those usually occur at coexistence lines between two phases. Second-order phase transitions occur at a critical point, in which the second derivative of the free energy is discontinuous. These usually occur at the end of coexistence lines. At the second-order phase transitions, i.e. at the critical point, the system displays interesting behaviour, such that the correlation length ξ diverges or that some observables/physical quantities follow certain power laws, which only depend on the type of the critical point. One example of such a power law is

$$\xi \propto \left(\frac{T - T_C}{T_C}\right)^{\nu}, \tag{3.12}$$

where ξ describes the correlation length and ν is a so called critical exponent, whose value is determined by the universality class of the system. When looking at the Ising model one can identify the line along the axis of zero magnetization from a temperature of zero up to the so called critical temperature T_C, as the coexistence line that splits the regions. In these regions most of the spins are either oriented up or down. The second order phase transition is located at the point of critical temperature T_C and zero

magnetization, i.e. at the end of the coexistence line. The discontinuity in the second derivative of the free energy can be translated into a divergence of the heat capacity. From this we can conclude that internal energy, whose derivative is heat capacity, has a pole at the critical point. This in turn means that when calculating the radius of convergence of the internal energy at a given temperature one should get the distance to the critical point.

However, the critical point does not exist on lattices with finite sizes, i.e. in performed simulations. This can be justified by the fact that for finite lattices the partition function is a finite sum of exponential functions. Thus it is always positive and the logarithm as well as its derivatives in the definition of the internal energy e are always continuous. Therefore there can be no phase transitions with finite volume and real T. However, if the T can be complex, the partition function may become zero and the logarithm as well as the internal energy may get poles, so called Fisher zeros. Due to the introduced singularities, the radius of convergence is finite, i.e. the distances to the Fisher zeros. We can therefore measure the position of the Fisher zeros in the complex plane (of the inverse temperature) by using different radii of convergence from different betas and triangulating the position.Adding this to the fact that the (leading) Fisher zeros converge towards the critical point, following the relationship from Equation 3.13 where β denotes the location of a Fisher Zero [9], [10], we can again calculate the critical point β_c, as well as the critical exponent ν from the radius of convergence.

$$|\beta - \beta_c| \propto L^{-1/\nu} \tag{3.13}$$

Formulated more precisely: As we know from the underlying physics that the critical point β_c has no imaginary part, we can calculate the critical exponent ν from the following equation

$$\mathrm{Im}\{\beta\} \propto L^{-1/\nu}. \tag{3.14}$$

Then using the obtained ν and Equation 3.13 we can determine β_c. It should be noted that in addition to the critical inverse temperature β_c, the critical exponent ν is also known. Its value for the two-dimensional Ising model is 1.

4 Numerical algorithms

This chapter will explain the fundamental numerical algorithms used and show the accompanying Julia code. For the Markov chain Monte Carlo (MCMC) simulation of the Ising model, as described in section 3.2, both the Metropolis algorithm and the Wolff algorithm are implemented. However only the Wolff algorithm is used in the final simulations, while the Metropolis algorithm was only used to verify the implementation of the Wolff algorithm. This chapter also shows the bootstrap algorithm used to calculate the errors and the algorithm used to calculate the derivatives on the grid.

But before we look at the individual algorithms, we need to look at how we can measure observables based on a simulation in the first place. Using Equation 3.2 in addition to the assumption that a given configuration occurs with probability p_i, leads to Equation 4.1 [11]

$$\langle O \rangle = \frac{\sum_i O_i \, p_i^{-1} \, e^{-\beta E_i}}{\sum_i p_i^{-1} \, e^{-\beta E_i}} \tag{4.1}$$

However, if in the MCMC simulation we require that the probability of a configuration is equal to the Boltzmann weight, the equation simplifies and we are left with a simple sum as in Equation 4.2.

$$\langle O \rangle = \frac{1}{N} \sum_i^N O_i \tag{4.2}$$

4.1 Wolff algorithm

The Wolff algorithm is implanted as described by Ulli Wolff in his original paper from 1988 [12]. The correctness of the algorithm, i.e. the fulfillment of the four MCMC conditions [11], [13] (positivity, normalization, ergodicity and (detailed-)balance) can be seen in the original paper from Ulli Wolff. The following code snippet Listing 4.1 performs a elementary cluster update step and is the main routine, closely following the four step process, a) to d), from Ulli Wolff.

Listing 4.1 Simple implementation of the Wolff cluster update, based on and closely following the method described in the original paper [12].

```
function WolffStep!(currentConfig,beta)
    #Initialization of variables
    Nx,Ny  = size(currentConfig)
    marked = zeros(Bool,Nx,Ny)
    queue  = Array{CartesianIndex{2},1}(undef,0)
    #Starting the cluster
    start = CartesianIndex(rand(rng,1:Nx),rand(rng,1:Ny))
    push!(queue,start)
    currentConfig[start] = -currentConfig[start] #flip the starting point
    marked[start] = true #mark the starting point

    # loop until the queue is empty
    while length(queue) != 0
        current_index = pop!(queue)
        for index in getNeighbors(current_index,Nx,Ny)
            if marked[index] continue end
            if currentConfig[current_index]==currentConfig[index] continue end
            P = 1-exp(-2*beta)
```

```
19              if rand(rng)<P
20                  currentConfig[index] = -currentConfig[index] #flip
21                  marked[index] = true #mark
22                  push!(queue,index) # add to queue
23              end
24          end
25      end
26      return sum(marked) #return clustersize
27  end
```

The following is an explanation of the functionality of the code with references to the four steps from the paper.

- (a) The first step involves selecting a starting point, which is performed on line 7.
- (b) Once a starting point is selected, it is flipped and marked as visited. This is done on lines 8 f.
- (c) The code then visits the nearest neighbors of the current index, attempting to activate a bond between the current index and its neighbors with probability P. During this process, the code visits the nearest neighbors of the current index, as done on line 15. If the bond between the current index and its neighbor is activated with probability P, the code flips the activated bond and marks it as visited, which is performed on lines 20f.
- (d) Finally, the code repeats steps (b) through (c) for any remaining unmarked neighbors until there are no more unmarked neighbors left to visit, which is done on line 13 by checking if the queue is empty

4.1.1 Optimization

Though the code from Listing 4.1 produces the proper results, it is quite slow. This can be traced back to two main reasons, the repeated memory allocation and the individual drawing of random numbers. In order to see how various changes improve performance a measure of the performance is needed. For this, the number of million spin flips per second (MFPS) is used. While it may not be the best measure, as it is depend on the lattice size and the inverse temperature of the simulation, it can be used to compare different algorithms when holding the aforementioned parameters constant. Using this metric, a baseline of (6.780 ± 0.019) MFPS can be established based on default code from Listing 4.1.

Regarding memory allocation, there are three obvious improvements that can be made. The marked, queue and neighbors array can be initialised at the beginning of the simulation instead of the start of every cluster update. The marked array will then only be needed to be reset at the beginning of a cluster update, i.e. filled with `false`. The neighbors array can simply be overwritten with the new neighbors of the current index. The queue list is replaced with a array of the maximum number of possible entries ($4 \cdot L^2$). Using this approach, a pointer is needed which keeps track of the current location and is increased or decreased when a value is added or removed to the queue, respectively. By using `track-allocation=user` as a argument when compiling and

executing the code a .mem file is created where the number at the beginning of each line denotes the assigned bytes for the respective line during the entire runtime of the program. The following Listing 4.2 shows the generated .mem file for an exert form the code of Listing 4.1.

Listing 4.2 Exert of a .mem file showing the total amount of assigned memory by line

```
1          0             P=1-exp(-2*beta)
2  103417152             if rand(rng)<P
3          0                 currentConfig[index]=-currentConfig[index] #flip
```

This shows that generating the random number for the accept-reject step is constantly reassigned. One can avoid this by initialising an array of length 1 and refilling it with a random number by using the `rand!(rng,r)` command which assigns the random numbers in place. The individual effect as well as the sum of these improvements, as in all implemented together, can be seen in Table 4.1.

Table 4.1 Speed measurements for the different optimised version of the Wolff algorithm. All runs have been performed using the following parameters: $L = 512$ $N_{meas} = 10\,000$ $\beta_0 = 0.435$ Independent runs $= 10$.
The Julia version used is 1.8.3 (2022-11-14).

changes w.r.t	absolute performance	relative improvement
baseline	(6.780 ± 0.019) MFPS	-
static queue array	(7.084 ± 0.034) MFPS	(1.05 ± 0.14)
static marked matrix	(7.085 ± 0.020) MFPS	(1.22 ± 0.11)
static neighbors array	(8.280 ± 0.042) MFPS	(1.41 ± 0.14)
static random array	(7.839 ± 0.097) MFPS	(1.25 ± 0.15)
all together	(12.034 ± 0.024) MFPS	(2.13 ± 0.20)

It shows that the changes to the overall memory allocation lead to an approximate doubling of the speed of the algorithm. Another improvement, in addition to the memory optimizations, can be achieved by changing the way random numbers are drawn. In the baseline version, Listing 4.1 as well as after the memory optimizations, random numbers are drawn individually when needed. By drawing a larger number of random numbers at once and regenerating them only when all have been used, the impact of the random number generator overhead can be reduced. Using this approach, the speed of the program reaches (16.43 ± 0.12) MFPS, which is a further increase of $(32.70 \pm 0.99)\,\%$ and results in a overall increase over the baseline code by a factor of (2.82 ± 0.26). The completely optimized code can be seen in Listing A.1

In addition to algorithmic optimisations, the nature of the underlying data, i.e. the way in which the spins are stored in memory, can also have an impact on the overall speed of the code. In Julia, there are four primitive types suitable for storing the spin(s): `Bool`, `Bitvector`, `Int` and `Float`. The `Bool` type stores a single `true` or `false` value. In theory, this would be the most suitable, since we only need to store whether the spin is up or down. However, the `Bool` type still takes up an entire byte instead of the one bit that is theoretically required. The type `Bitvector` solves this problem by

storing 8 `Bools` in one byte. This is the most compact way of storing the spins, but the compiler may not be able to optimise the code using this type, as it could with the `Bool` type. The `Int` is the most straight forward type, as it can directly store a 1 for spin up and a -1 for spin down. This allows, for example, to write the calculations used to compute the energy in a simpler and more convenient way. The type `Int` in Julia is implemented via the usable types `Int8`,`Int16`,`Int32` and `Int64`, which use the respective number of bits as specified in their names. Finally, `Float` (available via `Float16`, `Float32` and `Float64`) could also be used in the same way as `Int`. While it may seem strange to use `Float`, as floating-point arithmetic is significantly more complex than integer or boolean arithmetic, the increased number of FPUs, as opposed to ALUs, could compensate for this and result in an overall faster execution time. To determine the fastest type for this application, we again run multiple simulations using the optimised code from before for each type and determine the MFPS, the results can be seen in Table 4.2.

Table 4.2 Speed measurements for the different types used to represent the spin. All runs have been performed using the following parameters: $L = 512$ $N_{meas} = 10\,000$ $\beta_0 = 0.435$ Independent runs $= 10$. The Julia version used is 1.8.3 (2022-11-14)

Type used	absolute performance
Bitvector	(14.270 ± 0.018) MFPS
Bool	(16.411 ± 0.219) MFPS
Int8	(16.436 ± 0.047) MFPS
Int16	(16.359 ± 0.092) MFPS
Int32	(15.397 ± 0.026) MFPS
Int64	(15.111 ± 0.043) MFPS
Float16	(13.481 ± 0.035) MFPS
Float32	(15.667 ± 0.045) MFPS
Float64	(14.794 ± 0.077) MFPS

As can be seen, using `Int8` as the type gives the fastest execution time, with `Bool` also statistically matching the speed. Interestingly, `Bitvector` is the slowest type, probably due to the aforementioned reduced compile efficiency. Using `Float` is also definitely slower than using `Int8`, although interestingly `Float32` can compete with Int32. `Int8` is used for all following simulation as it is the fastest and its results in implementing certain functions whose readability is considerably greater than with `Bool`.

4.1.2 Sweep problem

Although the accompanying code of the cluster update has been omitted as it is not relevant for the algorithm, it is worth mentioning a small code piece that caused a significant amount of trouble.

Listing 4.3 Definition of a sweep function, which carries out a multiple iteration of the single Wolff update.

```
function WollfSweep!(currentConfig,Nx,Ny,beta)
    s=0::Int
    while s<Nx*Ny
        s+=WolffStep!(currentConfig,Nx,Ny,beta)
        # WolffStep!(...) returns the cluster size of the performed Wolff update
    end
    return s
end
```

The code in Listing 4.3 is supposed to alienated the problem of different autocorrelation length for different lattice sizes when the total number of measurements stays the same. Instead of calling `WolffStep!` one would call `WolffSweep!` which adjust the time between measurements by repeating n cluster updates until L^2 points have been flipped. In addition to the keeping the autocorrelation length constant, the code was also designed to help with comparing the autocorrelation with the metropolis simulation using a checkerboard sweep. While it has achieved the goal of consistent autocorrelation, with the continuum limit still matching Onsager's solution, see Figure 4.1, it changes the internal energy for small lattice sizes.

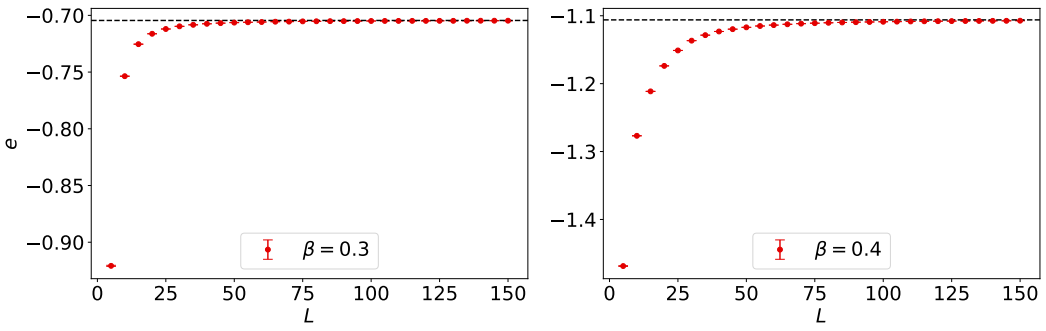


Figure 4.1 Plot of the internal energy e for increasing lattices sizes, from $L = 5$ up to $L = 150$, compared to the expected internal energy e calculated form the Onsager solution of the Ising model. The parameters of all the simulations are $N_{sep} = 0$ $N_{therm} = 40$ $N_{meas} = 1\,000\,000$.

To the see the effect on simulations using small lattices, the summation w.r.t the cluster sizes is increased form 1 to L^2 in increments of 1. Figure 4.2 shows the internal energy at a inverse temperature of 0.3 and $L = 2, 4, 6$ with the aforementioned increase in the summation limit compared to the internal energy calculated form the respective partition function.

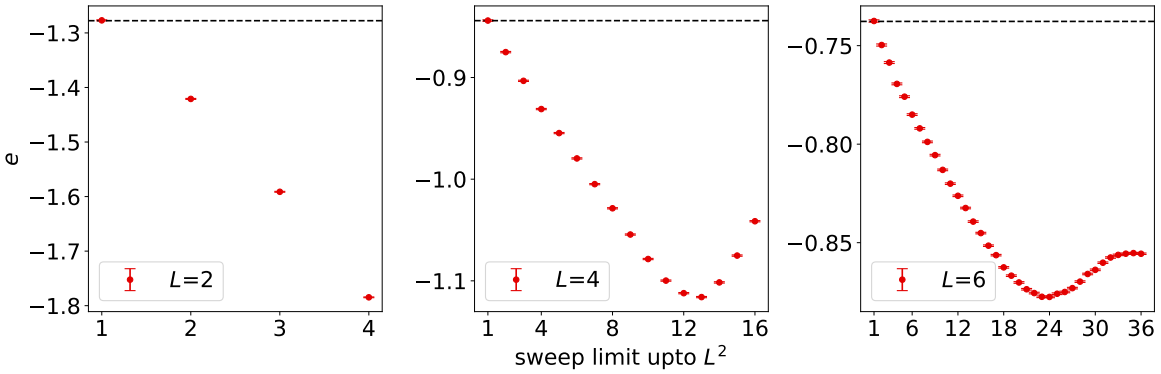


Figure 4.2 Plot of the internal energy e for a increasing summation limit inside the `WolffSweep!` function up to L^2 at a inverse temperature of 0.3 compared to the expected internal energy e calculated form the respective partition function. The parameters of all the simulations are $N_{sep} = 0$ $N_{therm} = 40$ $N_{meas} = 1\,000\,000$.

One can clearly see that for a limit of 1, which is equivalent leaving out the function call to `WolffSweep!`, because in the Wolff algorithm the starting spin is always filliped, the simulation agrees with the theoretically expected solution. However, as soon as the summation limit is increased, the energy begins to differs significantly, with the maximum deviation being $39.69\,\%$ for $L = 2$

The deviation is probably caused by the fact, that the introduction of the termination condition from the `WolffSweep!` function causes a bias towards measuring configurations which flip big number spins i.e. updates with bigger cluster sizes. Bigger cluster sizes in turn correlate with a bigger number of likewise spins, as only likewise spins can be combined in a cluster, and therefore with a lower energy, as one can deduce form Equation 4.3 where the Hamiltonian has been rewritten in terms of the (constant) number of possible pairs of neighbors N_N, number of aligned pairs of neighbors N_A and number of opposite pairs of neighbors N_O.

$$H = -J \sum_{<ij>} \sigma_i \sigma_j = -2JN_N + 4JN_O = -2JN_N + 4J(N_N - N_A) \tag{4.3}$$

$$= +2JN_N - 4J(\boldsymbol{N_A}) \tag{4.4}$$

4.2 Bootstrap

In order to calculate the statistical error on the observables, bootstrapping is used, which is a resampling algorithm derived from the Jackknife resampling algorithm [14]. The fundamental idea is to simply repeat the experiment (simulation) a given amount of times and then calculate the observable, one is interested in, on each experiment. The spread of all the resulting observables can then be used to calculate the error on the observable. However, instead of repeating the simulation a given number of times, as this would be clearly be too computational expensive, new simulation are generated based on the original simulation. This is done by sampling randomly from the original simulation, henceforth called the central sample, until we have a new sample. While this may seem counterintuitive at first, it turns out that this method can give accurate

estimates of the statistical error, especially in cases where the central sample is large enough and the underlying probability distribution is not too complicated. The error on the calculated error itself is then equal to the computed error divided by the square root of the number of Bootstrap samples, i.e. the number of repeated simulations. The number of Bootstrap samples one should use is a contested topic. Inside of this thesis 1000 number of samples are used as it is on the upper bound of the what is computational achievable with some of the samples that will be analysed and seems to be a acceptable number of samples. It is important to note that when analysing data in which each data point is not completely independent of the previous data point, i.e. the data are correlated, as is the case in MCMC simulations, an extra step must be introduced in order not to underestimate the error. This step is called blocking and is done before the bootstrap algorithm. The idea is to block the data and use these blocks as data points to resample. By doing this, it is ensured that the bootstrap samples will contain the same correlations as the central sample. This of course introduces another parameter, blocksize, which must be tuned alongside the number of bootstrap samples. Although the block size can be tuned manually by looking at the bootstrap error and increasing the block size until the error stops growing, a good approximation is $2\tau^{int}+1$, where τ^{int} is defined as the integrated autocorrelation time. See [13] for an explanation of the integrated autocorrelation time and its calculation. The code used to carrying out the Bootstrap algorithm is displayed in Listing A.2

4.3 Recursive differentiation algorithm

In order to calculate the derivatives of observables O, such as e or m, using the simulation based on the MCMC, the formula for recursive differentiation, as described in subsection 3.1.1, needs to be implemented. For this, a general derivative function for the **n**th derivative is implemented which recursively calls itself and caches already computed expectation values and lower derivatives.

First, we need a way of tracking the expectation value and distinguishing between cases such as $\langle H^2\rangle$ and $\langle H\rangle^2$. To do this, we use the struct `OH` to keep track of the number of H and O in an expectation value, like this: $\langle H^n \cdot O^m\rangle$. Additionally, to keep track of the nth derivatives of an expectation value, a struct `D` is used. The definition of these structures can be found in Listing 4.4.

Listing 4.4 Structs used for save the the Julia function to recursively calculate the derivative of with regards to a observable O consisting in the form of OH^n, where H is the external energy

```
struct OH # eq. to <O^n*H^m> with n stored in OH.O & m stored in OH:H
    O::Int #Number of occurences of the observable in the EV
    H::Int #Number of occurences of the energy in the EV
end
struct D # eq. to d^n/d beta^n <obs>
    n::Int # n'th derivetive
    obs::OH # of the obs
end
```

Using the structs, we can define the function `calcDiff`, which creates a dictionary

to cache the values of the derivatives, and define a function `EV`, which is used to compute the expectation values needed during the calculation of the derivatives. The function `EV`, as defined in Listing 4.5 from lines 5 to 13, will also cache the individual expectation values (EV) and calculate them if they have not already been calculated using the bootstrap method. It is important to note that although the resampling itself is completely random, the same resampling order is used for all the different expected values. The defined function then calls a function at the end which, using the defined caches and the method for calculating the EV, can recursively calculate the desired derivative.

Listing 4.5 Julia function to kickstart the calculation of the derivative a observable using bootstraping.

```
1  function calcDiff(observable,H,n,weights,blocksize,calcEV=nothing,calcD=nothing)
2      # initializing calcEV & calcD if not provided
3      # left out for readability
4      ...
5      function EV(obs::OH)
6          if haskey(calcEV,obs)
7              return calcEV[obs]
8          else
9              obs_array=observable.^obs.O .* H.^obs.H
10             m=bootstrap_all(obs_array,weights,blocksize)
11             calcEV[obs]=m
12             return m
13         end
14     end
15     return d(OH(1,0),n,EV,calcD)
16 end
```

The called function `d(...)`, whose full definition can be seen in Listing 4.6, either returns the EV if n=0 or calculates the derivative using the formula

$$\partial_\beta \langle O \rangle = - \langle EO \rangle + \langle O \rangle \langle E \rangle .$$

It does this by starting with the expression for the first derivative. For higher derivatives, the first term can be easily differentiated using the recursive formula. For the second term, the general Leibniz rule is needed to split the product so that the recursive formula can be applied. Thus, all derivatives of order n can be explicitly traced back to order $n-1$, on which `d(...)` can be recursively called until $n = 0$.

Listing 4.6 Julia function to recursively calculate the derivative of with regards to a observable O consisting in the form of OH^n, where H is the external energy

```
1  function d(obs::OH,n,EV,calcD)
2      if haskey(calcD,D(n,obs)) return calcD[D(n,obs)] end
3      if n==0
4          return_val = EV(obs)
5      else
6          m=n-1
7          s=sum(
8              binomial(m,k)*d(obs,m-k,EV,calcD).*d(OH(0,1),k,EV,calcD)
9          for k=0:m)
10         return_val=-1*d(inc(obs),n-1,EV,calcD).+s # inc(obs) : obs.H => obs.H+1
11     end ; calcD[D(n,obs)]=ret_val ; return ret_val
12 end
```

5 Results

This chapter summarises all the results obtained using the simulation code and the analysis method derived in the previous chapters. In particular, it will show the intermediate results obtained up to the final results for the critical exponent ν and the critical inverse temperature β_c. At the beginning we will also check the underlying simulations and some calculated secondary observables. At the end an analysis of the systematic error is carried out.

Unless stated otherwise, all calculations, including the final evaluation of the partition function, are done using double-precision binary floating-point math. The statistical errors of all simulated results are calculated using the bootstrap method, see section 4.2, where the block size is set to $2\tau_{int}+1$ and the number of bootstrap samples is 1000. The simulations themselves are performed using the Wolff algorithm, see section 4.1. For each respective Plot N_{sep}, N_{therm}, N_{meas} are stated. N_{sep} denotes the number of cluster updates between the measurement of the observables. N_{therm} denotes the number of thermalization updates to reach the thermal equilibrium. It should be stated that for each set of simulation, the thermalization has been checked and sometimes the number of thermalization updates has been increased vastly over the required amount. Finally N_{meas} denotes the total number of recorded samples for each observable (internal energy and magnetization).

5.1 Verification of the simulation

Before looking at the resulting data and the associated analysis, we will examine the simulated data and compare it to the known values. This is done for the very small lattice sizes, $L = 2, 4, 6$, where the simulated results can be compared directly with the derived expression based on the partition function. In addition, the simulated data are compared to the Onsager solution of the Ising model. This is done by looking at the data of simulations with increasing lattice sizes, up to $L = 100$, in order to extrapolate the continuum limit.

5.1.1 Verification for small lattice sizes

To verify the simulation for small lattice sizes, the exact partition functions as calculated in section 3.2 are used. The Onsager solution of the Ising model is not applicable due to finite volume effects, see subsection 3.2.1.
For $L = 2, 4, 6$ the partition functions are evaluated at seven inverse temperatures up to the critical inverse temperature. These values are compared with the simulations carried out at the respective lattice sizes and temperatures. The following Figure 5.1 shows the deviation of the internal energy in percent.

Figure 5.1 shows that for small L the simulation agrees with the analytically expected value. The maximum deviation is less than 0.021 %; the corresponding reduced chi-squared statistics of $2.55, 0.85, 2.65$ are also acceptable.

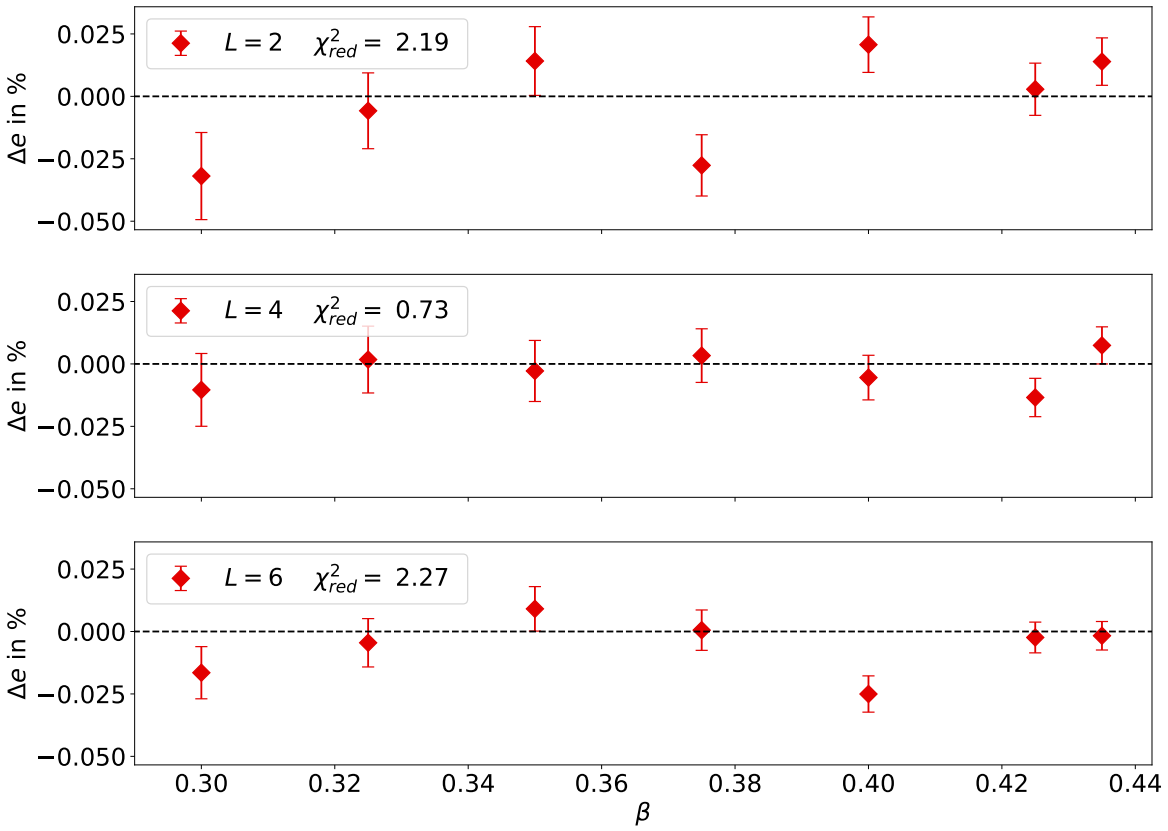


Figure 5.1 Comparison of the internal energy between simulation and analytical result for $L = 2, 4, 6$. The parameters of all the simulations are $N_{sep} = 200$ $N_{therm} = 4\,000$ $N_{meas} = 20\,000\,000$.

5.1.2 Verification for big lattice sizes

To verify the correctness for larger lattice sizes, up to 100^2, one has to take the Onsager solution as an analytical result. However, this leads to a significant discrepancy for smaller L, because the finite volume effects are still significant. Nevertheless, the comparison can show whether the simulation approaches the continuum limit, i.e. if the simulated results converge correctly.

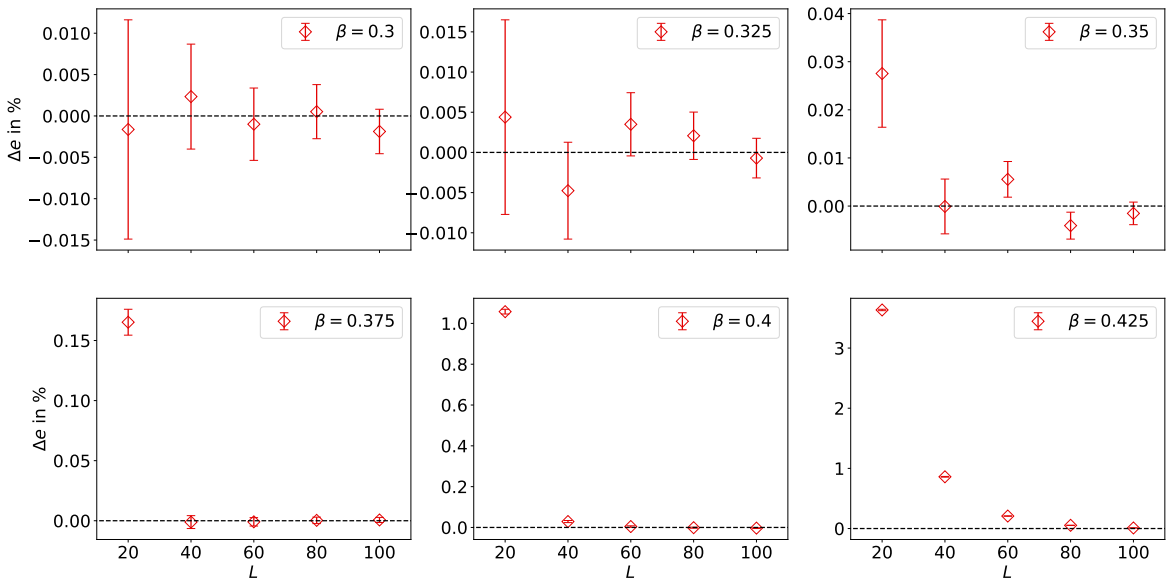


Figure 5.2 Comparison of the internal energy between the simulation and the Onsager solution. The parameters of all the simulations are $N_{sep} = 2000$ $N_{therm} = 40$ $N_{meas} = 1\,000\,000$.

Figure 5.2 shows that for each inverse temperature examined, the calculated internal energy converges to the analytically expected one, with the rate of convergence decreasing as the inverse temperature approaches the critical inverse temperature.

5.2 Derivative

Having verified the underlying simulation, we will now look at the derivatives of the internal energy and compare them with the analytical result. The comparison will focus on the inverse temperatures of 0.3 and 0.4 in order to have an analysis near and far from the critical point where the rates of convergence are different. Derivatives are calculated using the algorithm described in section 4.3. The parameters of all the simulations performed in this section are $N_{sep} = 2000$, $N_{therm} = 4000$, $N_{meas} = 1\,000\,000$ unless stated otherwise.

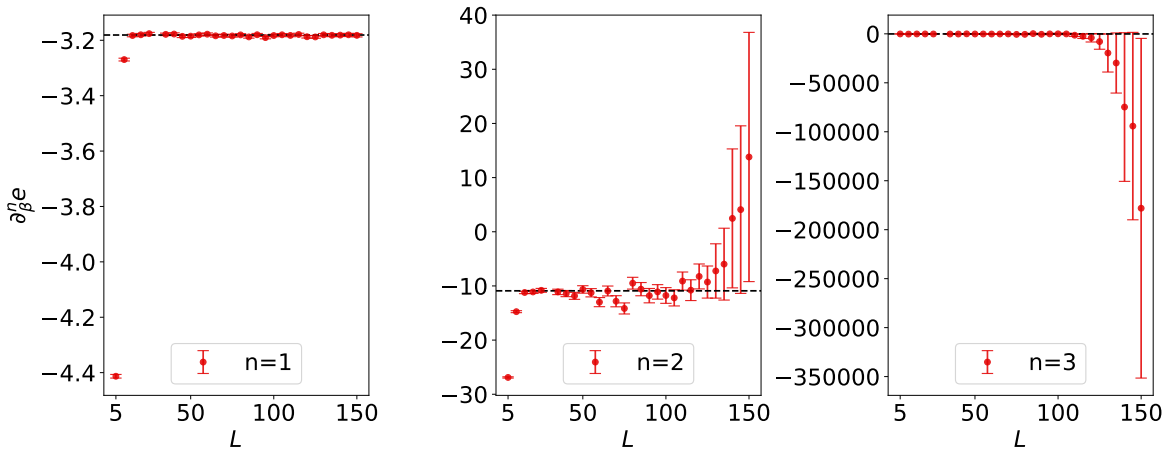


Figure 5.3 Comparison of the first three derivatives of the internal energy between the simulation and the Onsager solution, which is indicated by the dotted black line, at $\beta = 0.3$.

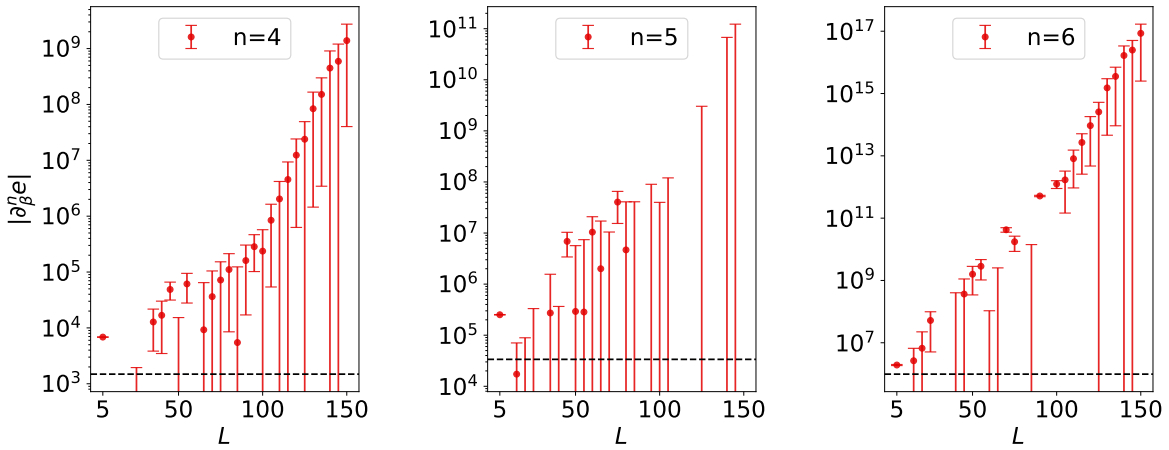


Figure 5.4 Comparison of the fourth to sixth derivative of the internal energy between the simulation and the Onsager solution with a logarithmic scale,at $\beta = 0.3$.

5.2.1 Derivative on large lattices

We will first look at the derivatives for larger lattices, up to 150^2. Calculating the first three derivatives for a range of lattices to see any convergence or divergence.

As one can see in Figure 5.3 for an inverse temperature of 0.3, the derivative converges quickly with respect to L to the Onsager solution. However, the 2^{nd} and 3^{rd} derivatives already start to diverge again for lattice sizes of around 100 and larger. By looking at higher derivatives and using a logarithmic scale for the absolute value of the derivative, the divergence can clearly be seen, see Figure 5.4.

If we look at the same graphs for a β value of 0.4, see Figure 5.5, we do not observe any divergence for the first three derivatives. The third derivative data points, though, already show a significant amount of noise.

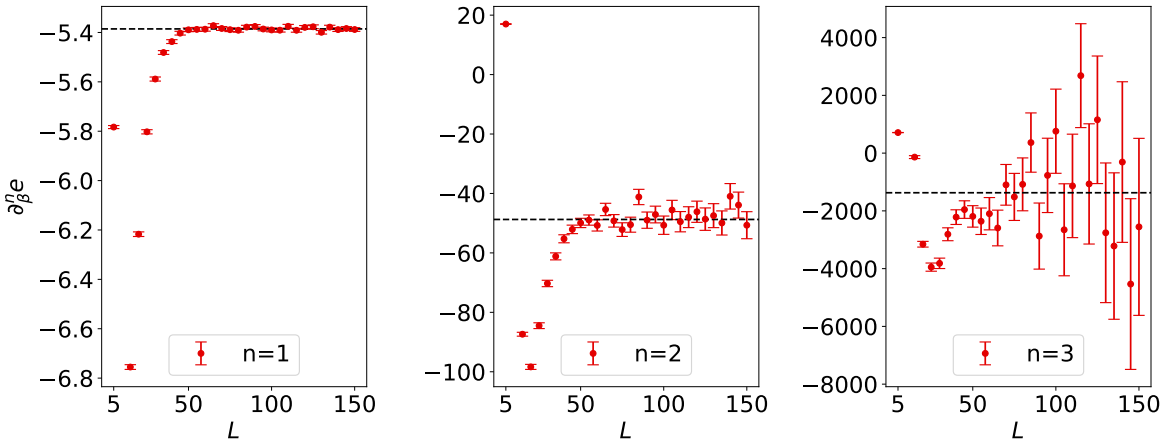


Figure 5.5 Comparison of the first three derivatives of the internal energy between the simulation and the Onsager solution, at $\beta = 0.4$.

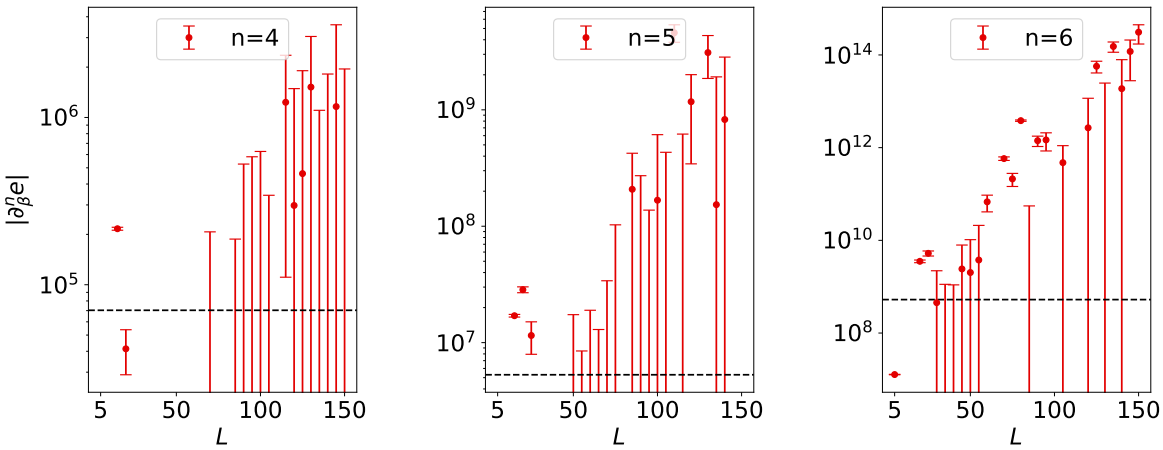


Figure 5.6 Comparison of the fourth to sixth derivative of the internal energy between the simulation and the Onsager solution with a logarithmic scale,at $\beta = 0.4$.

When we look at the higher derivatives with a logarithmic scale, however, we see that the derivative diverges again, as can be seen in Figure 5.6.

The divergence can be explained by looking at the definition of the various derivatives. We know from the Onsager solution that the internal energy derivatives are intensive observables because the values of the derivatives are finite. However, in the definition of derivatives on a finite lattice (see subsection 3.1.1), extensive observables appear, such as $\langle H \rangle$ or $\langle H^2 \rangle$, which tend to infinity as L increases. We therefore have a sign problem, i.e. we take the difference of values that increase in size, while expecting the difference to be finite, resulting in less accuracy as L increases. This explains the divergence for bigger L as well as the accompanying increase in the errors of L. To confirm that this is the case and not the result of an error in the code of the recursion algorithm used to compute the derivatives, as described in section 4.3, we can compare the results with those obtained using the symbolic equations. Figure 5.7 and Figure 5.8 show the results of the derivatives using the formulas from subsection A.1.1.

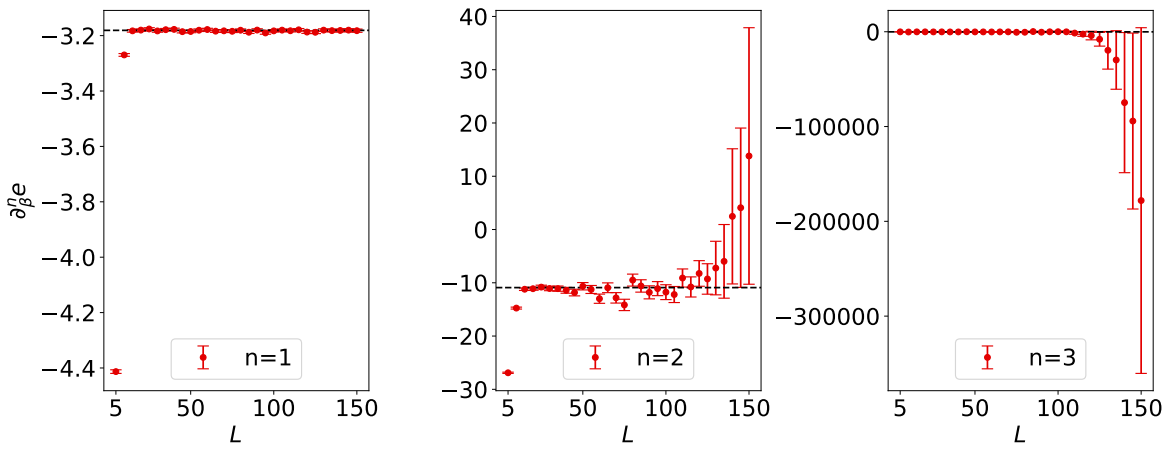


Figure 5.7 Comparison of the first three derivatives of the internal energy between the simulation and the Onsager solution, at $\beta = 0.3$. The derivatives are calculated using the symbolically derived equations

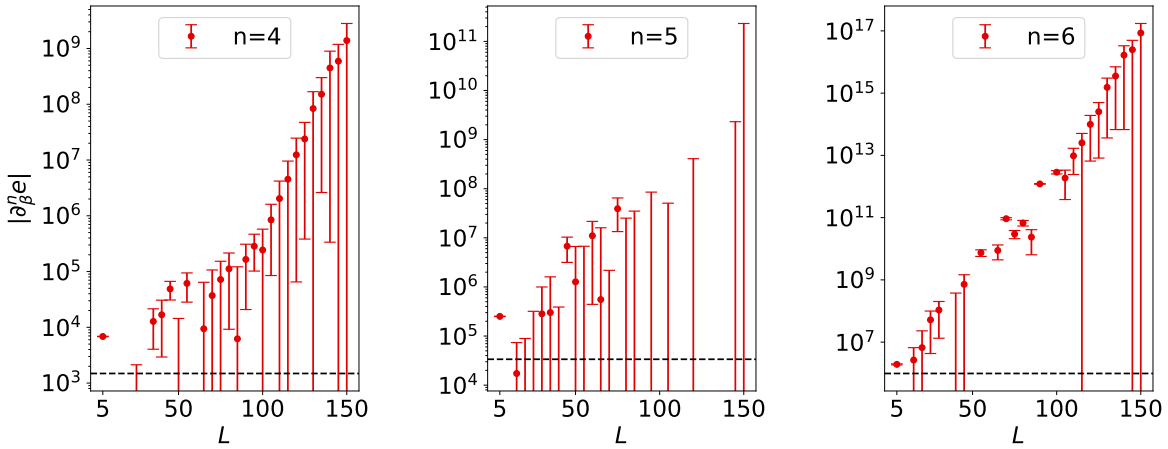


Figure 5.8 Comparison of the fourth to sixth derivative of the internal energy between the simulation and the Onsager solution with a logarithmic scale, at $\beta = 0.3$. The derivatives are calculated using the symbolically derived equations.

These figures show, especially when compared to Figure 5.3 & Figure 5.4, that the divergence still occurs and verify the correctness of the underlying differentiation algorithm.

5.2.2 Derivative on small lattices

To circumvent the sign problem, we calculate the derivative on smaller lattices, since we have already seen that the simulation itself gives correct values for the internal energy, and try to extrapolate to larger lattice sizes. In addition, for the small lattice sizes for which it is feasible to compute the partition function, we can compare the results with those obtained from the partition function. Figure 5.9 and Figure 5.10 are such a comparisons for an inverse temperature of 0.3 and 0.4, respectively.

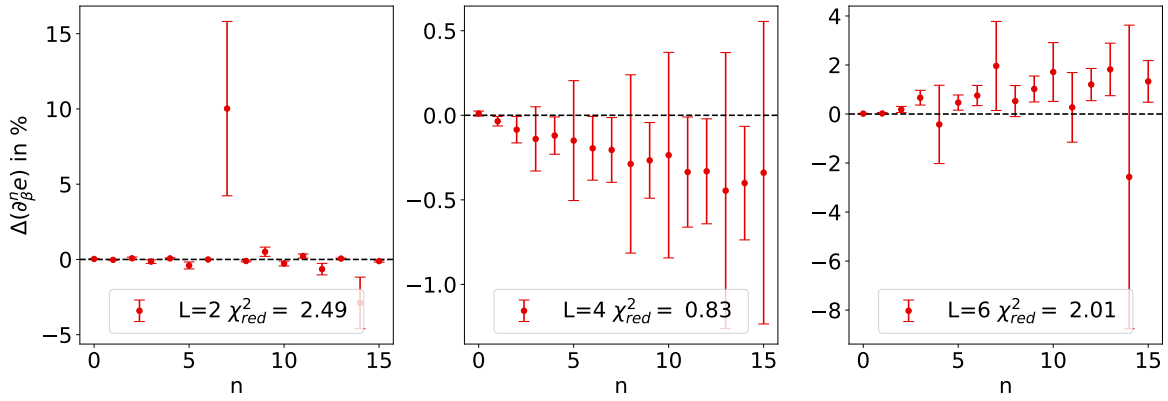


Figure 5.9 Comparison of the derivatives for the lattice sizes of $L = 2, 4, 6$ to the analytic results obtained from the respective partition function at $\beta = 0.3$.
The parameters of all the simulations are $N_{sep} = 200$ $N_{therm} = 4000$ $N_{meas} = 20\,000\,000$.

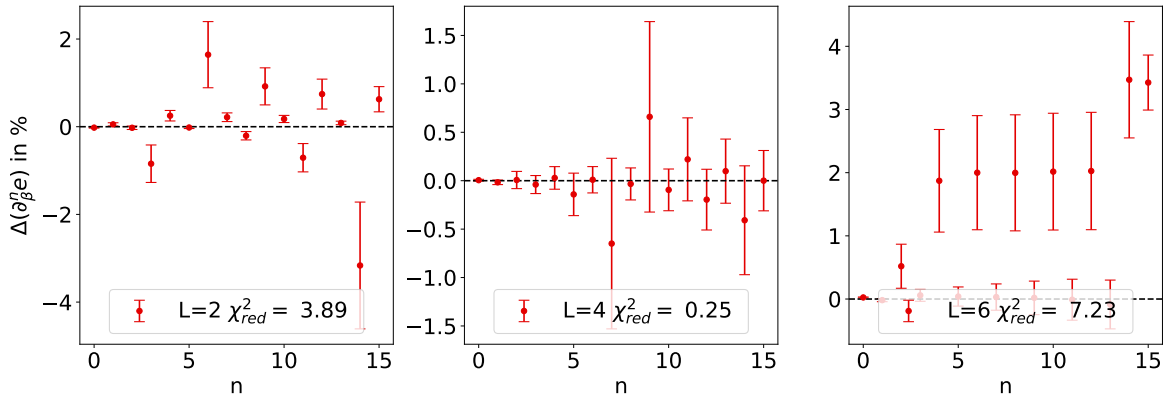


Figure 5.10 Comparison of the derivatives for the lattice sizes of $L = 2, 4, 6$ to the analytic results obtained from the respective partition function at $\beta = 0.4$.
The parameters of all the simulations are $N_{sep} = 200$ $N_{therm} = 4000$ $N_{meas} = 20\,000\,000$.

More comparisons for inverse temperatures of $\beta = 0.325, 0.35, 0.375, 0.425, 0.435$, can be seen in section A.3. In general, the respective χ^2_{red} indicated that the derivatives measured on the simulations were in agreement with the theoretically expected ones. It should nevertheless be emphasized that there are significant numbers of outliers,

especially when including the plots in the appendix, which however always have a correspondingly high error. A significant divergence, as in the case of the derivation based on the large lattices, cannot be determined in any case. Perhaps an investigation with a higher number of bootstrap samples than 1000, e.g. 10000, would be appropriate to reduce the increased χ^2.

5.3 Radius of convergence

Now that we can calculate derivatives of the internal energy e, albeit for small lattices, the knowledge from chapter 2 can be used to calculate the radius of convergence of $e(\beta-\beta_0)$ at a specific β_0. We will first compare the two methods available for calculating the radius of convergence, the square root test and the ratio test. We then determine the convergence radius by estimating the limit of the respective infinite series. The parameters of all following simulations are $N_{sep} = 200$ $N_{therm} = 4000$ $N_{meas} = 20\,000\,000$.

5.3.1 Comparison of the root and ratio test

Both methods of calculating the convergence radius require the determination of a limit where the n^{nth} elements of the sequence are related to the nth or (n)^{nth} plus 1 derivative, respectively. The following Figure 5.11 is a direct compression of the first 50 elements in the respective series of the methods for a lattice size of $L = 6$ and an inverse temperature of $\beta = 0.3$. In these and all subsequent Figures, r_n is the value of the n^{th} elements of the respective series.

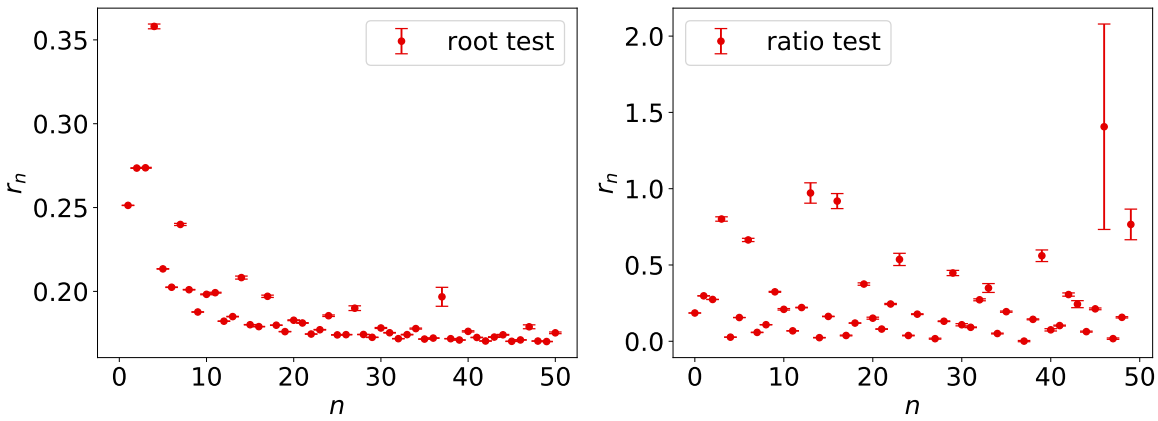


Figure 5.11 Comparison of the root and ratio test for $L = 6$ & $\beta = 0.3$ up to the first 50 elements.

As can be seen, the sequence elements of the root test converge quickly, in contrast to the results of the ratio test, where the spread is so large that no conclusion can be drawn about the convergence. Although many more plots can be created in regards to the comparisons, they all lead to the same result, as stated above. One thing to note is that there can sometimes be oscillations in the derivatives, see Figure 5.12 where $L = 12$ & $\beta = 0.425$. While the root method dampens these, especially when correctly applying the limit inferior, the ratio method behaves strangely as it takes only the

ratios into account. A demonstration of both methods is shown in Figure 5.12 [1] Taking into account the damping of the oscillations as well as the generally better convergence, the root test seems to be better suited to determine the radius of convergence.

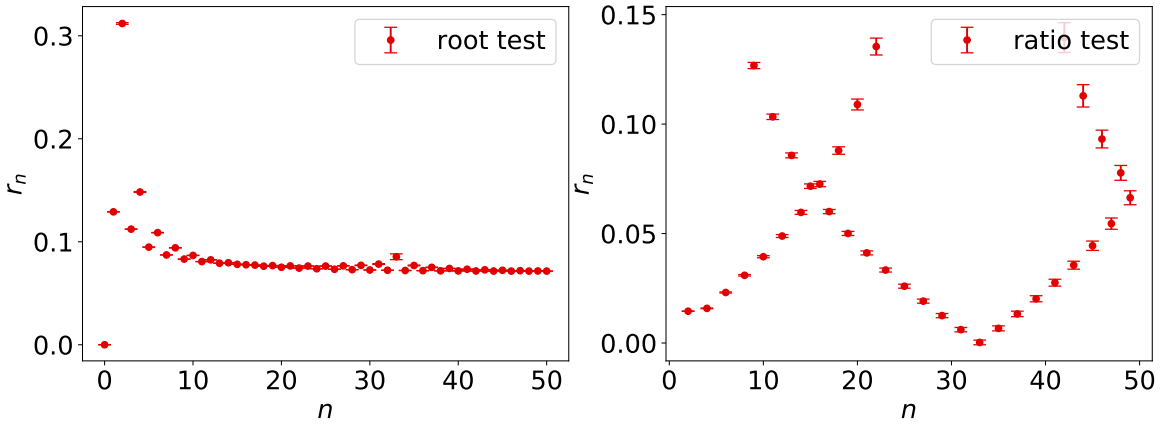


Figure 5.12 Comparison of the root and ratio test for $L = 11$ & $\beta = 0.4$ up to the first 50 elements.

While we have already see the derivatives diverge for large lattice, the divergence could in theory cancel out in the ratio test or impact the root test negligible. Applying the root and ratio test a simulation performed at a lattice size of $L = 100$ and at a critical inverse temperate of β_c, however results in a radius of convergence, form the root and ratio test near zero, see Figure 5.13. The expected value, based on the known critical inverse temperature is however $\beta_c - 0.3 \approx 0.14$. A small deviation would be expected, as the Fisher zeros could still be approaching the critical point, but such a large deviation is not consistent with the theory.

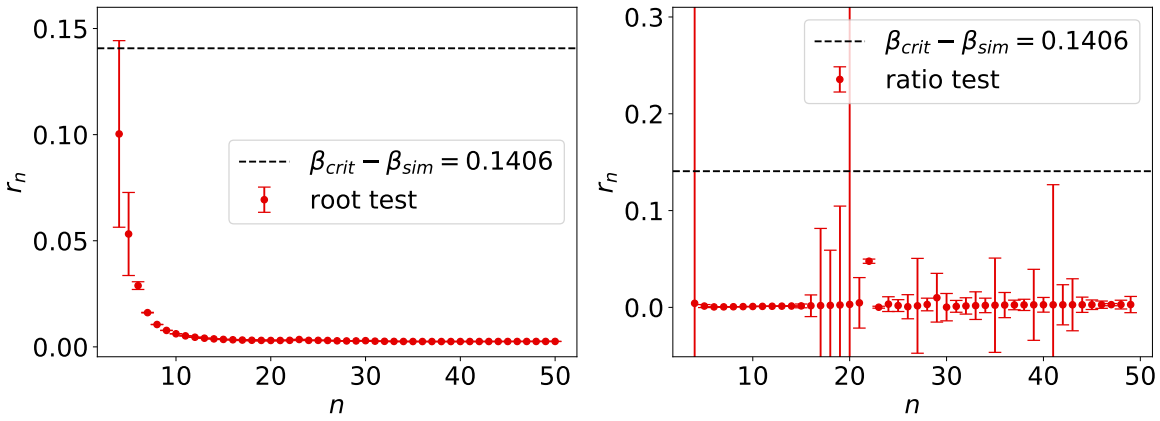


Figure 5.13 Comparison of root and ratio test for $L = 100$ & $\beta = 0.3$ up to the first 50 elements. The black dotted line shows the approximate expected radius of convergence based on the expansion to the critical point.

[1] Here one can see the difference between the *limit* and the *limit inferior*. While no limit might exist for certain series, the limit inferior always exists.

5.3.2 Determining the limit

In order to extract the actual convergence radius from the series generated by the root test, it is necessary to extrapolate to infinity, as only a finite number of sequence elements can be calculated. To do this, the series elements must be described by a function from which the limit (-inferior) can be determined. Since the shape of the convergence in the root test is not yet known, no function can be chosen for the fit based on the underlying information or physical properties of the system. It is therefore necessary to simply select a function and check that it fits the data points. One of the simplest functions that can model a general decay, as seen in the root test above (Figure 5.11 & Figure 5.12) with convergence to a constant value is $f(n) = a + b/n$. While more complicated functions may be used, we will focus on the reciprocal function with an added constant for now. Applying the fit function to the series elements from Figure 5.11, where the first 10 elements of the series have been omitted to improve the convergence of the fit, gives the result that can be seen in Figure 5.14. While the

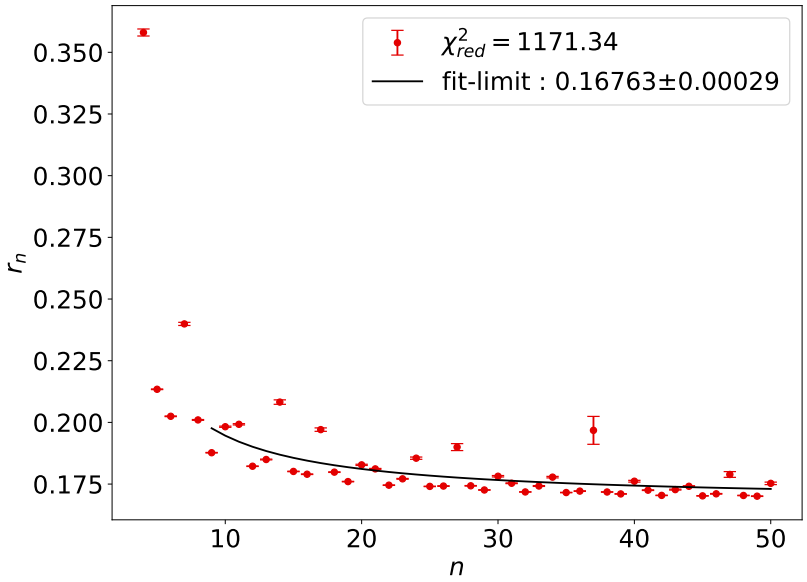


Figure 5.14 Fit with the fit-function $f(n) = a + b/n$ of the elements of the series (starting at $n = 10$) created by the root test for the radius of convergence at $L = 6$ & $\beta = 0.3$.

fit is visually consistent with the data points, the χ^2_{red} of 1171.34 suggests that the fit function does not describe the data points well.However, this is to be expected as the fit function should only be an approximation of the shape of the data points as explained above. The χ^2_{red} can be improved by including the inferior in the definition of the square root test. Although we cannot reproduce the exact behavior of the limit inferior without knowing all the infinite elements in the series, we can transform the series into a strictly monotone decreasing one and use it to find the limit. This can be achieved by discarding any element in the series greater than its preceding element. Figure 5.15 shows such a monotonic series with an accompanying fit. Although one can clearly see an improvement, which is consistent with the reduced χ^2_{red} of 15.07,

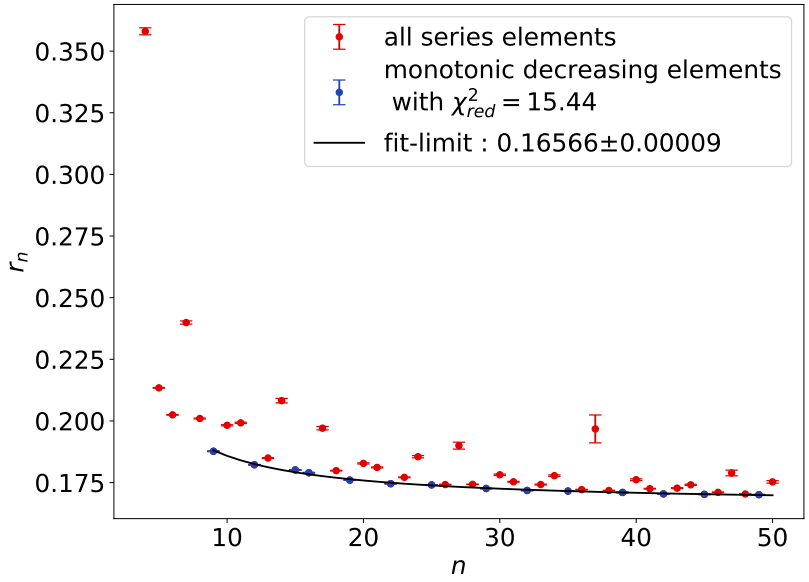


Figure 5.15 Fit with the fit-function $f(n) = a + b/n$ of the strictly monotonic decreasing elements of the series (starting at $n = 10$) created by the root test for the radius of convergence at $L = 6$ & $\beta = 0.3$.

the χ^2_{red} is still too high to conclude that the fit function is accurately describing the elements of the series. However, we will stick to the fitting function $f(n) = a + b/n$ in conjunction with the condition for strictly monotone decreasing series elements to estimate the radius of convergence. Carrying out the same procedure for multiple inverse temperatures, ranging from 0.3 to 0.44, results in Figure 5.16. As can be clearly seen, the convergence radii are different for each beta. Looking closely at the colours, we can see that the convergence radius starts to decrease monotonically as the inverse temperature increases from 0.3 to 0.4, while it begins to increase again for the inverse temperatures of 0.425, 0.435 and 0.44. Repeating the same procedure for lattice sizes from $L = 2$ to $L = 13$ leads to similar plots, which are included in the Appendix, see section A.4. It should also be noted that since the infimum method failed to produce a monotonically decreasing sequence for $L = 2$, the data from the $L = 2$ simulation will be omitted from further analysis. For the reason see, subsection A.4.1.

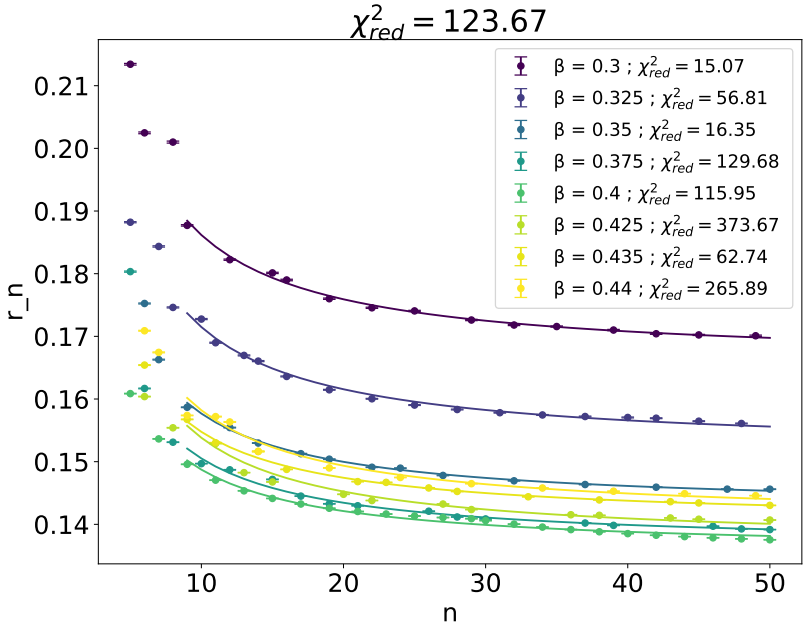


Figure 5.16 Multiple fits for the radius of convergence at $L = 6$ and inverse temperatures ranging from 0.3 to 0.44. All fit functions used are of the form: $f(n) = a + b/n$ with the requirement that the series be monotonically decreasing, as explained above. The χ^2_{red} are computed for each individual fit as well as for the combination of all fits and data points.

5.4 Determining the Fisher Zeros

Now that we have estimated the convergence radius for individual lattice sizes and inverse temperatures, albeit with too large a value of χ^2_{red}, we can use these to determine the location of the Fisher zeros, as described in subsection 3.2.1. Figure 5.17 shows the limits calculated on the basis of Figure 5.16, i.e. the different radii of convergence, drawn as circles, representing the domain on which the internal energy converges. The centre of the circle is therefore the point at which the simulation is carried out, while the radius is equal to the radius of convergence. In addition, the point of intersection of all the edges of the circles is shown, which is determined by the minimisation of the least squares of the distances. Figure 5.17 shows a clear point at which all the circles meet, i.e. a point by which all the radii of convergence are bound. The same process converges successfully for all simulated lattice sizes except for $L = 12$ and $L = 13$, due to the fact that the calculated radii of convergence lead to an arrangement of circles for which no clear intersection can be determined. See Figure 5.18 for an example.

While the minimisation algorithm can of course determine the point with the minimum distance to the circle edges, it is visually clear that the resulting point does not represent a Fisher zero. This is probably due to the sign problem already affecting the results. The Fisher zeros resulting from $L = 12$ and $L = 13$ are therefore ignored.

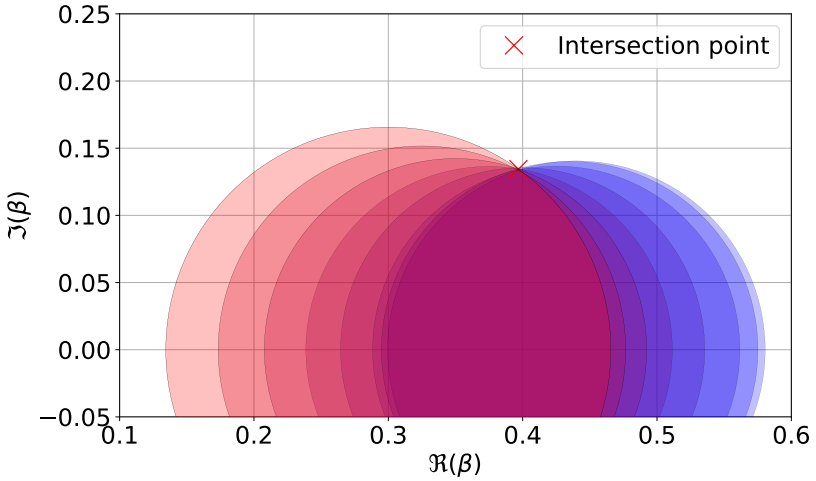


Figure 5.17 Overlay of several convergence circles based on the results from Figure 5.16 which are based on a lattice size of $L = 6$ and inverse temperatures ranging from 0.3 up to 0.44, resulting in a Intersection point of $\beta = 0.39684(10) + i\,0.13436(2)$.

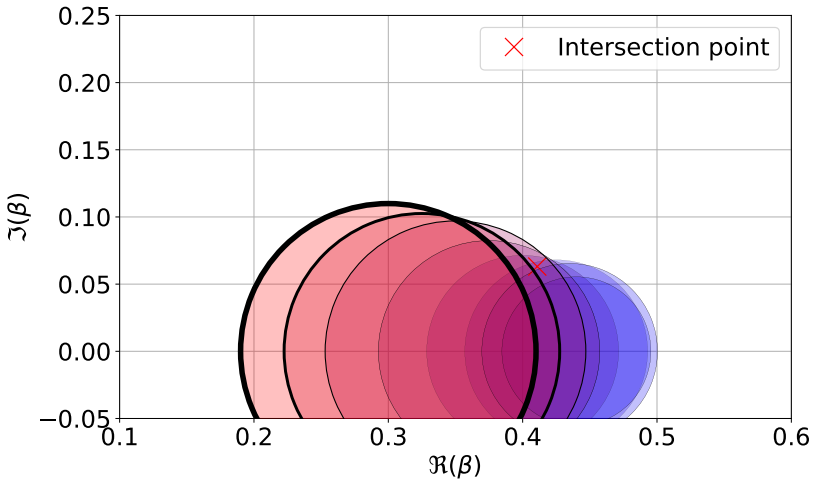


Figure 5.18 Overlay of several convergence circles based on the results from simulations performed at lattice size of $L = 12$.

5.4.1 Comparison to Deger & Flindt

Before we use the scaling behaviour of the Fisher zeros to determine the critical point in conjunction with the critical exponent, we compare the locations of the Fisher zeros with known values. For this, we use the data from the already cited paper [10] by Deger & Flindt, who kindly provided the calculated positions of the Fisher zeros for this comparison. Figure 5.19 shows a side-by-side comparison of the Fisher zeros computed so far, as well as those of Deger & Flindt, and the location of the critical point in the thermodynamic limit. Grid sizes smaller than $L = 5$ have been omitted because Deger & Flindt did not compute the location of the Fisher zeros for such small grids. In addition, we can only check the match visually, since the errors calculated so far are too small, as we can see from χ^2_{red}, which are significantly larger than 1. While

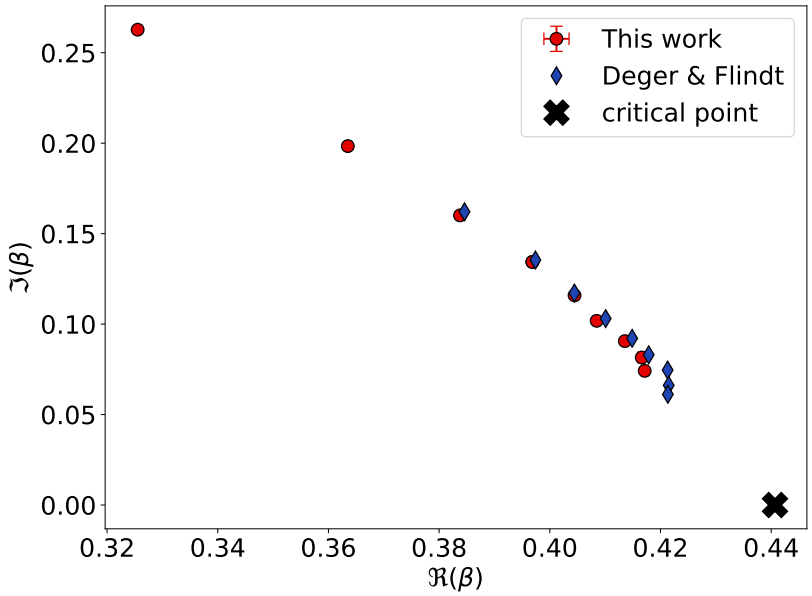


Figure 5.19 Comparison of the Fisher zeros in the inverse temperature plane with the position provided by Deger & Flindt [10].

Figure 5.19 roughly shows the approximate agreement, for a better overview Figure 5.20 plots the respective imaginary and real parts of the Fisher zeros. The analysis shows that there is good agreement between the imaginary parts of the Fischer zeros and the comparison data. In addition, the real parts are in agreement with the exception of a significant deviation observed for a lattice size of 11. However, it should be noted that this agreement only concerns the behaviour of the Fisher zeros. Taking into account the statistical error calculated so far, there is no agreement.

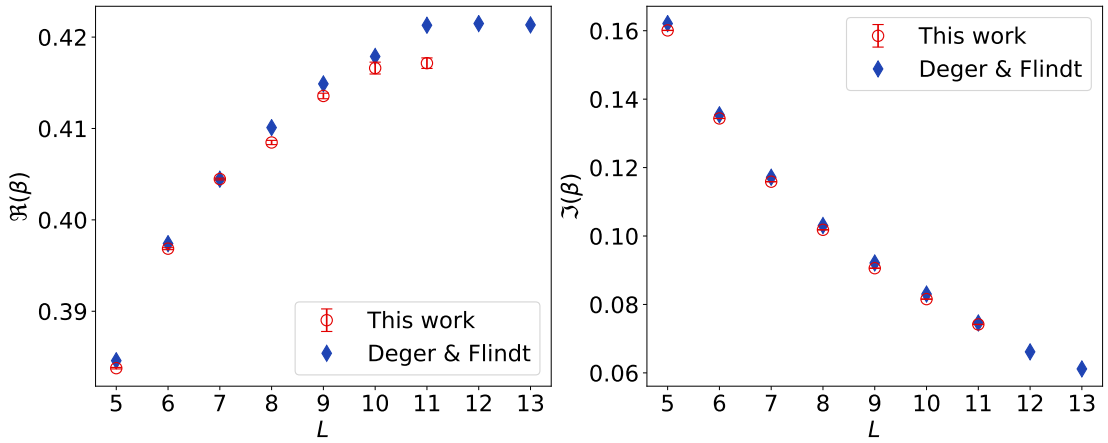


Figure 5.20 Individual comparison of the real and imaginary parts of Fisher zeros with Deger & Flindt's Fisher zeros [10].

5.5 Determining the critical exponent ν and inverse temperature β_c

Using the determined imaginary and real parts of the Fisher zeros, in conjunction with the scaling as explained in subsection 3.2.1, we can compute the critical exponent ν and the inverse temperature β_c. Instead of determining ν by fitting the exponential scaling function $\mathrm{Im}\{\beta\} \propto L^{-1/\nu}$ directly, we determine the slope s in the double-log plot, ν is then given by $-1/s$. Figure 5.21 shows the described double log plot with the calculated ν.

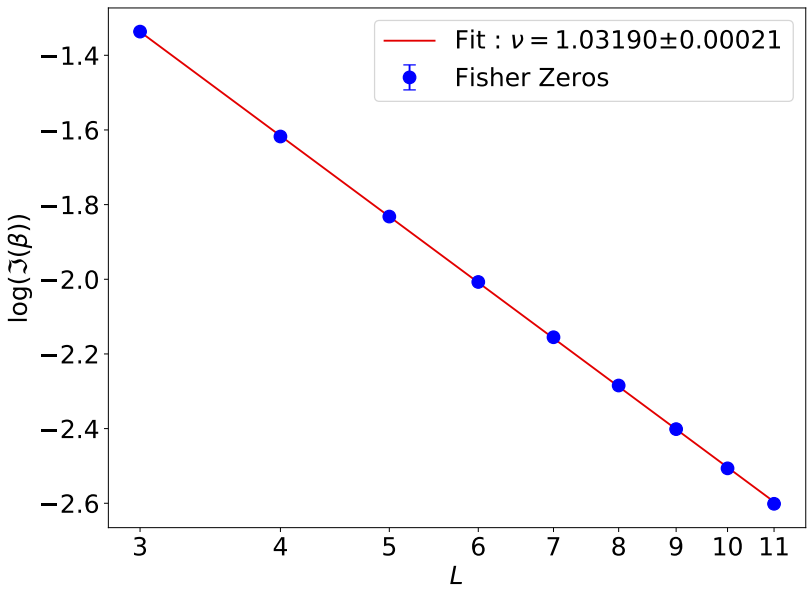


Figure 5.21 Figure showing the fit to extract the critical exponent ν from the scaling behaviour of the identified Fisher zeros, resulting in $\nu = 1.031\,90 \pm 0.000\,21$.

The slope $s = -0.969\,08 \pm 0.000\,19$ results in a value for ν of $1.031\,90 \pm 0.000\,21$. Using the estimated value of the critical exponent ν, we can now extract the critical inverse temperature by using the scaling relation from subsection 3.2.1:

$$|\beta - \beta_c| \propto L^{-1/\nu}$$

and a linear fit. The extrapolated value of β for $L^{-1/\nu} = 0$ then corresponds to β_c, see Figure 5.22 for the fit.

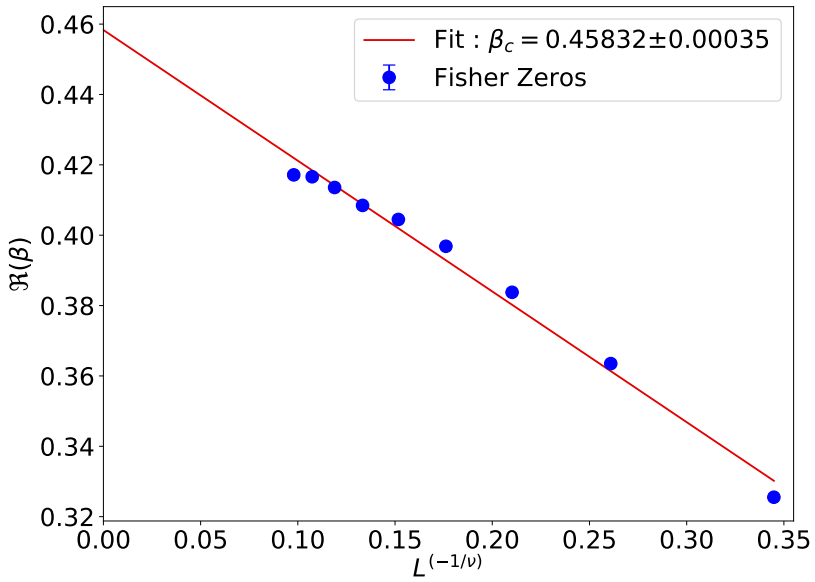


Figure 5.22 Figure showing the fit to extract the critical inverse temperature β_c using the critical exponent ν from Figure 5.21 and the scaling behaviour of the identified Fisher zeros, resulting a y-intercept of $0.458\,32 \pm 0.000\,35$.

The resulting critical inverse temperature β_c is equal to $0.458\,32 \pm 0.000\,35$. The calculated ν differs from the theoretical value by 3.1%, while β_c is off by 4.1%. Taking into account the calculated statistical error, one can see that the values are not comparable with the theoretical values. A possible reason could be that the systematic error dominates the total error instead of the statistical error. The error analysis continues in section 5.6 where the total error is determined.

5.5.1 Determination of ν & β_c using the magnetization

In addition to the internal energy e, the magnetisation m, defined as the sum of the spins divided by the volume, is another primary observable of the Ising model. So far we have only used the internal energy e as a basis for the determination of the Fisher zeros. However, since the magnetisation is also defined by a derivative of the logarithm of the partition function[2], its radius of convergence could also be used to derive the Fisher zeros.

[2]Definition of the magnetization using the partition function: $M(B,T) = 1/(L^2)\partial_B \ln Z((B,T)$ where B is a external magnetic field. This definition holds true even if B is subsequently set to zero.

Carrying out the previous analysis with the magnetization m as the observable results in a critical exponent of $\nu = 1.060\,63 \pm 0.020\,07$ and a critical inverse temperature of $\beta_c = 0.444\,55 \pm 0.007\,31$. The associated fits for ν and β_c can be seen in Figure 5.23, while the plots containing the radii of convergence and Fisher zeros can be found in section A.6.

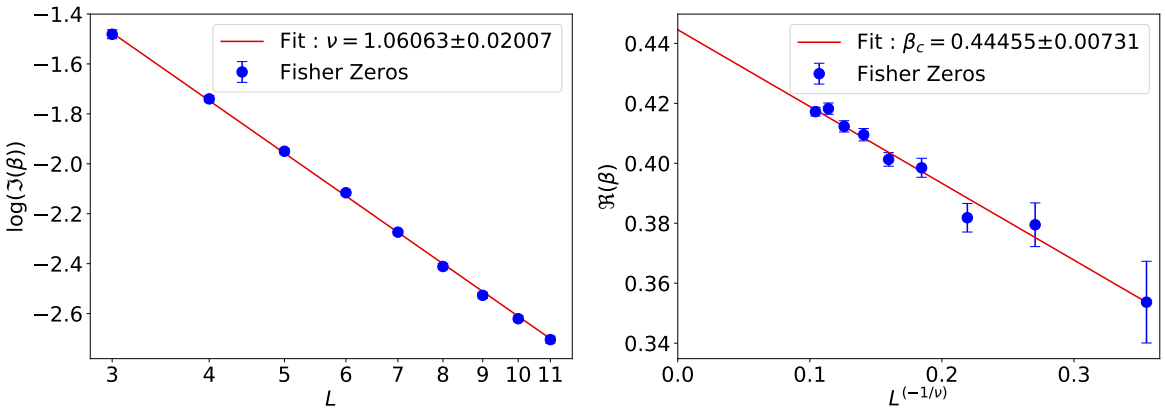


Figure 5.23 Determination of the critical exponent $\nu = 1.060\,63 \pm 0.020\,07$ and critical inverse temperature of $\beta_c = 0.444\,55 \pm 0.007\,31$.

The increased statistical error is a result of the magnetisation itself having a larger error than the internal energy. While ν is still more than three standard deviations away from the analytical value, the calculated β_c agrees with the theory within one standard deviation.

5.6 Determination of the total error

As we have seen in the previous section 5.5, the statistical error cannot be used to conclude that the calculated critical exponent ν and inverse temperature β_c agree with the analytical results, when using the internal energy e as a basis for the radii of convergence. A possible explanation could be that the total error of these observations is dominated by the systematic error, which is not included in the discussion of the results. To verify this possibility, the determination of the total error will be the subject of this section. This is done by calculating ν and β_c based on a number of different fit functions and examining the resulting distribution. More specifically, the convergence radii on which ν and β_c are based are determined using different fit functions, different fit ranges and with or without the limit inferior correction. All resulting values for ν and β_c are then combined into a histogram. This histogram can then be transformed into a Cumulative distribution function (CDF) from which the total error can be calculated. For a thorough discussion of this method, see [3].

In addition to the reciprocal function previously used with an added constant, nine other functions are introduced, all of which have a limit of zero, such that these functions combined with a constant term can be used to describe the sequence generated by the root test. The functions consist of six polynomials in $1/x$, one exponential decay model, two logarithmic functions and a simple constant function. A list of the functions

used can be found in section A.7. In addition, the start of the fit range is varied from five to forty, resulting in 36 different fit ranges. Finally, the fits are performed once with and once without the monotonicity condition. All this results in 720 different fit methods that can be used to determine ν and β_c. In theory, each entry in the histogram should be weighted according to the Akaike Information Criterion (AIC) to avoid over-representation of certain fits that do not describe the data. However, due to the increased χ^2_{red} of all the fits, all the ACI weights except the one corresponding to the lowest χ^2_{red} are zero. We therefore have to resort to flat weighting, i.e. assigning the same weight to each fit model. As a side effect, flat weighting tends to produce the largest error. The total error is therefore unlikely to be underestimated. It should be emphasised that these models can only estimate the systematic error introduced by the fit of the radii of convergence. They do not help in the determination of the systematic errors introduced by the calculation of the Fisher zeros or those systematic errors introduced by the final fits for ν and β_c. In addition, clearly non-converging models are omitted from the histogram, e.g. $\nu < 0$ or those with degrees of freedom less than 1. This was necessary because, as discussed earlier, flat weighting cannot filter these out because it does not take goodness of fit into account. Figure 5.24 displays the normalised histogram, generated using the steps described above, for the critical exponent ν with flat weighting.

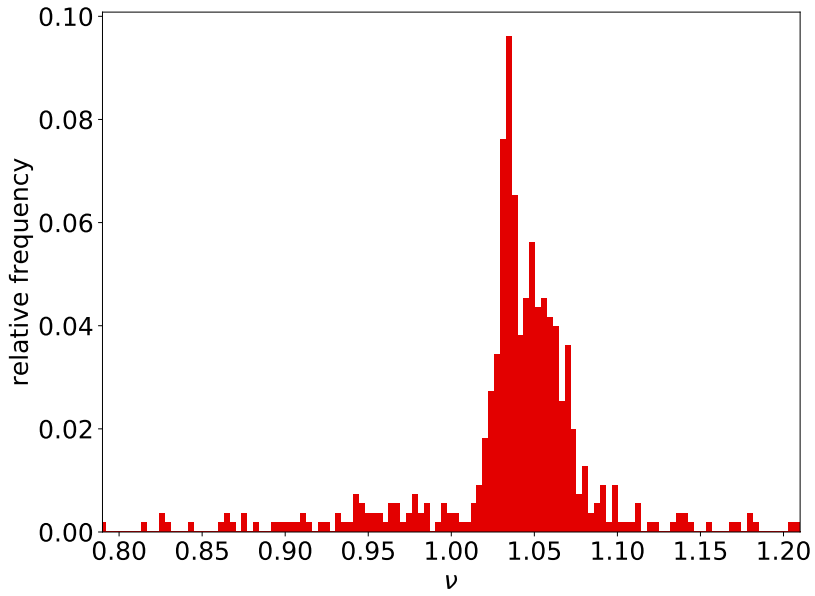


Figure 5.24 Figure showing the normalised histogram containing all the calculated critical exponents ν based on 720 different fitting models.

The cumulative sum of the individual bins of the histogram gives the corresponding CDF, which can be seen in Figure 5.25.

Based on the values of ν, where the CDF reaches plus and minus one standard deviation ($\approx 0.50 \pm 0.34$) from the median, we can give the critical exponent with the total errors to be $\nu = 1.0399^{+0.0283}_{-0.0389}$. We can see that the difference in terms of standard deviations is 1.02 compared to the theoretical value of 1.

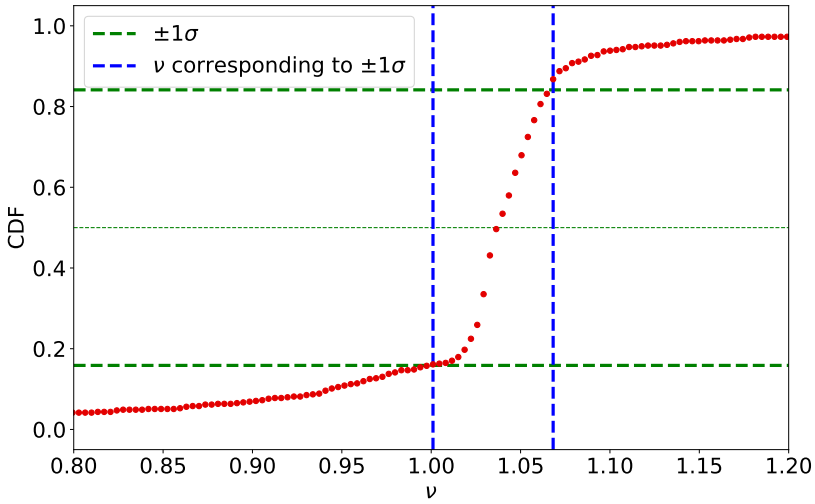


Figure 5.25 Figure showing the CDF of Figure 5.24. The median is at $\nu = 1.0399$ with the one sigma deviation to the right at $\nu = 1.0682$ and to the left at $\nu = 1.0010$.

For the critical inverse temperature, both the normalised histogram and the CDF are given in Figure 5.26.

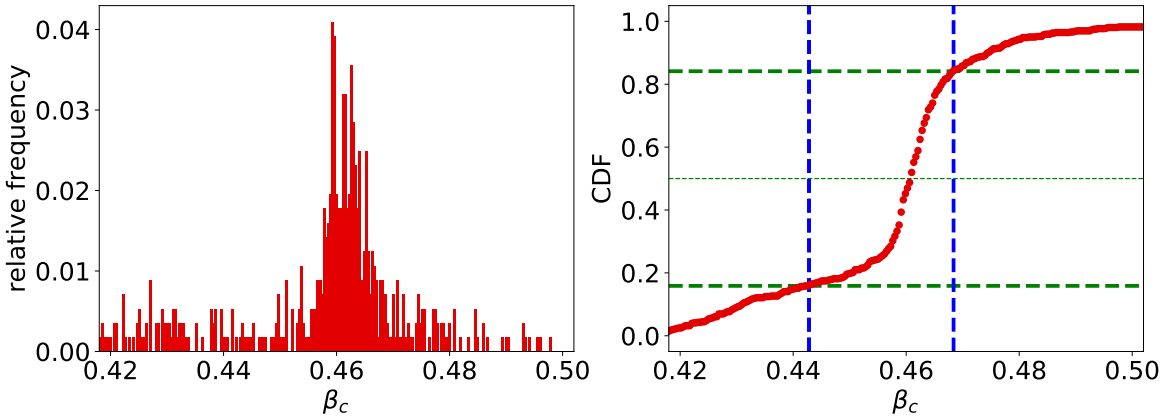


Figure 5.26 Figure showing the normalised histogram as well as the CDF for the critical inverse temperature. The median is at $\beta_c = 0.4609$ with the one sigma deviation to the right at $\beta_c = 0.4685$ and to the left at $\beta_c = 0.4428$.

The final estimated value of the critical inverse temperature with the total error is $\beta_c = 0.4609^{+0.0076}_{-0.0180}$. The calculated β_c differs from the theoretical value by about 1.12 standard deviations.

6 Conclusion and outlook

In this thesis, the method for determining the critical point using the radius of convergence was implemented. Simulations of the two-dimensional Ising model were successfully performed using the Wolff algorithm.

For large lattices, such as $L = 100$, the method fails to determine the critical point. The failure of the method can be traced back to the values of the derivatives used to determine the radii of convergence which begin to diverge as the lattice size increases. The failure of the method is due to the values of the derivatives which begin to diverge as the lattice size increases. Those derivatives cause the radii of convergence to tend towards zero. The observed divergence is probably due to a sign problem introduced in the numerical calculation of the derivatives.

However, for small lattices, L up to 11, we found that the derivatives could be computed successfully, as the sign problem, while still present, can be overcome by appropriately increasing the statistic. Following the aforementioned method, the radii of convergence were calculated and the Fisher factors were estimated by finding the intersection points of the resulting radii of convergence. Taking the imaginary parts of the Fisher zeros in conjunction with the described finite volume scaling, the critical exponent is calculated to be $\nu = 1.0399^{+0.0283}_{-0.0389}$. Using the computed value for the critical exponent ν and the real parts of the Fisher zeros, the critical inverse temperature is determined to be $\beta_c = 0.4609^{+0.0076}_{-0.0180}$. The errors are calculated by analysing the distribution of the respective values of ν and β_c generated by using different fitting models. The values of critical exponent ν is compatible within 1.02 σ with the value predicted by the theory, while the result for β_c is compatible within 1.12 σ with the Onsager solution for the critical point. In conclusion, the method is suitable for the determination of the critical point in two dimensional Ising model, although the error is relatively large.

As an outlook, the estimation of the total error should also include those generated by finding the intersection of the radii of convergence and the extrapolation of ν as well as β. One could then try to reduce the systematic error, as it is clearly the dominant error in the total error. This could be achieved by using a more appropriate range of fitting models, preferably choosing one based on a mathematical formulation of the shape of convergence, to find a model best suited for estimating the limit. In addition, applying the method to more complex models could provide a better understanding of the method and help to improve it.

A Appendix

A.1 Derivatives on the lattice

The second order derivative carried out by hand.

$$\frac{\partial^2}{\partial\beta^2}\langle O\rangle = \frac{\partial}{\partial\beta}(-\langle EO\rangle + \langle O\rangle\langle E\rangle) \quad = -\frac{\partial}{\partial\beta}(\langle EO\rangle) + \frac{\partial}{\partial\beta}(\langle O\rangle\langle E\rangle) \tag{A.1}$$

$$= -[-\langle HHO\rangle + \langle HO\rangle\langle H\rangle] + (-\langle HH\rangle + \langle H\rangle\langle H\rangle)\langle O\rangle \tag{A.2}$$

$$+ (-\langle HO\rangle + \langle O\rangle\langle H\rangle)\langle H\rangle \tag{A.3}$$

$$= \left\langle H^2O\right\rangle - 2\langle HO\rangle\langle H\rangle + 2\langle H\rangle^2\langle O\rangle - \left\langle H^2\right\rangle\langle O\rangle \tag{A.4}$$

A.1.1 Symbolically calculated derivatives

The following are the first six derivatives of the expected value of the energy with respect to β.

$$\frac{\partial^0}{\partial\beta^0}\langle E\rangle = \langle E\rangle \tag{A.5}$$

$$\frac{\partial^1}{\partial\beta^1}\langle E\rangle = \langle E\rangle^2 - \left\langle E^2\right\rangle \tag{A.6}$$

$$\frac{\partial^2}{\partial\beta^2}\langle E\rangle = 2\langle E\rangle^3 - 3\langle E\rangle\left\langle E^2\right\rangle + \left\langle E^3\right\rangle \tag{A.7}$$

$$\frac{\partial^3}{\partial\beta^3}\langle E\rangle = 6\langle E\rangle^4 - 12\langle E\rangle^2\left\langle E^2\right\rangle + 4\langle E\rangle\left\langle E^3\right\rangle + 3\left\langle E^2\right\rangle^2 - \left\langle E^4\right\rangle \tag{A.8}$$

$$\frac{\partial^4}{\partial\beta^4}\langle E\rangle = 24\langle E\rangle^5 - 60\langle E\rangle^3\left\langle E^2\right\rangle + 20\langle E\rangle^2\left\langle E^3\right\rangle + 30\langle E\rangle\left\langle E^2\right\rangle^2 - 5\langle E\rangle\left\langle E^4\right\rangle - 10\left\langle E^2\right\rangle\left\langle E^3\right\rangle + \left\langle E^5\right\rangle \tag{A.9}$$

$$\frac{\partial^5}{\partial\beta^5}\langle E\rangle = 120\langle E\rangle^6 - 360\langle E\rangle^4\left\langle E^2\right\rangle + 120\langle E\rangle^3\left\langle E^3\right\rangle + 270\langle E\rangle^2\left\langle E^2\right\rangle^2 - 30\langle E\rangle^2\left\langle E^4\right\rangle - 120\langle E\rangle\left\langle E^2\right\rangle\left\langle E^3\right\rangle + 6\langle E\rangle\left\langle E^5\right\rangle - 30\left\langle E^2\right\rangle^3 + 15\left\langle E^2\right\rangle\left\langle E^4\right\rangle + 10\left\langle E^3\right\rangle^2 - \left\langle E^6\right\rangle \tag{A.10}$$

$$\frac{\partial^6}{\partial\beta^6}\langle E\rangle = 720\langle E\rangle^7 - 2520\langle E\rangle^5\left\langle E^2\right\rangle + 840\langle E\rangle^4\left\langle E^3\right\rangle + 2520\langle E\rangle^3\left\langle E^2\right\rangle^2 - 210\langle E\rangle^3\left\langle E^4\right\rangle - 1260\langle E\rangle^2\left\langle E^2\right\rangle\left\langle E^3\right\rangle + 42\langle E\rangle^2\left\langle E^5\right\rangle - 630\langle E\rangle\left\langle E^2\right\rangle^3 + 210\langle E\rangle\left\langle E^2\right\rangle\left\langle E^4\right\rangle + 140\langle E\rangle\left\langle E^3\right\rangle^2 - 7\langle E\rangle\left\langle E^6\right\rangle + 210\left\langle E^2\right\rangle^2\left\langle E^3\right\rangle - 21\left\langle E^2\right\rangle\left\langle E^5\right\rangle - 35\left\langle E^3\right\rangle\left\langle E^4\right\rangle + \left\langle E^7\right\rangle \tag{A.11}$$

A.2 Optimization

Listing A.1 Final form of the implantation of the Wolff algorithm

```
function WolffStep!(currentConfig,beta,queue,marked,neighbors,r,r_idx)
    fill!(marked, false)
    Nx,Ny  = size(currentConfig)
    start = CartesianIndex(rand(rng,1:Nx),rand(rng,1:Ny))
    q_idx=1
    queue[q_idx]=start
    currentConfig[start]=-currentConfig[start] #flip
    marked[start]=true #mark
    while q_idx != 0
        current_index=queue[q_idx]
        q_idx-=1
        updateNeighbors!(neighbors,current_index,Nx,Ny)
        for index in neighbors
            if marked[index] continue end
            if currentConfig[current_index]==currentConfig[index] continue end
            P=1-exp(-2beta)
            r_idx[1]=r_idx[1]+1
            if r_idx[1]>length(r)
                rand!(rng,r) #reset random numbers
                r_idx[1]=1  #reset idx numbers
            end
            if r[r_idx[1]]<P
                currentConfig[index]=-currentConfig[index] #flip
                marked[index]=true #mark
                q_idx+=1
                queue[q_idx]=index
            end
        end
    end
    return (marked)
end
```

Listing A.2 The functions used to calculate the bootstrap samples, generate the resampling weights and blocking of observables.

```
function bootstrap_all(Observable,weights,blocksize)
    blockedObservable=blocking(Observable,blocksize)
    all_means=zeros(eltype(blockedObservable),length(weights))
    for i=1:length(weights)
        all_means[i]=StatsBase.mean(blockedObservable,weights[i])
    end ; return all_means
end
function genRandomWeights(N,l)
    weights=[FrequencyWeights(ones(Int8,l))] ; sizehint!(weights,N+1)
    for i=1:N
        push!(weights,genRandomWeightsVector1D(l))
    end ; return weights
end
function blocking(observable::AbstractArray,blocksize::Int)
    new_length=floor(Int,length(observable)/blocksize)
    result=zeros(eltype(observable),new_length)
    for i=1:new_length
        result[i]=StatsBase.mean(observable[(i-1)*blocksize+1:(i)*blocksize])
    end ; return result
end
```

A.3 Additional comparisons for the derivative of the internal energy at small lattice sizes

The following are comparison of the derivatives of the internal energy for the lattice sizes of $L = 2, 4, 6$ to the analytic results obtained from the respective partition function at $\beta = 0.325, 0.35, 0.375, 0.425, 0.435$.

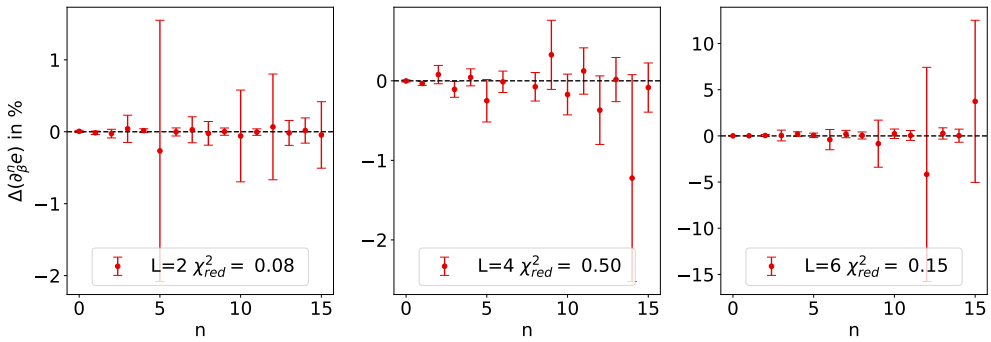


Figure A.1 $\beta = 0.325$

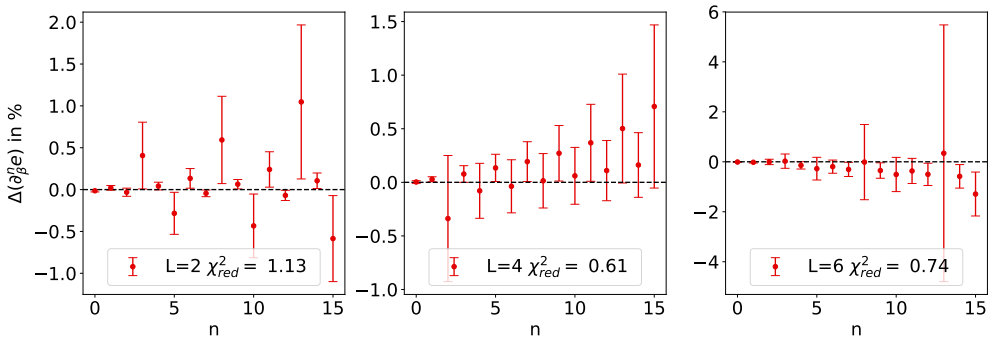


Figure A.2 $\beta = 0.35$

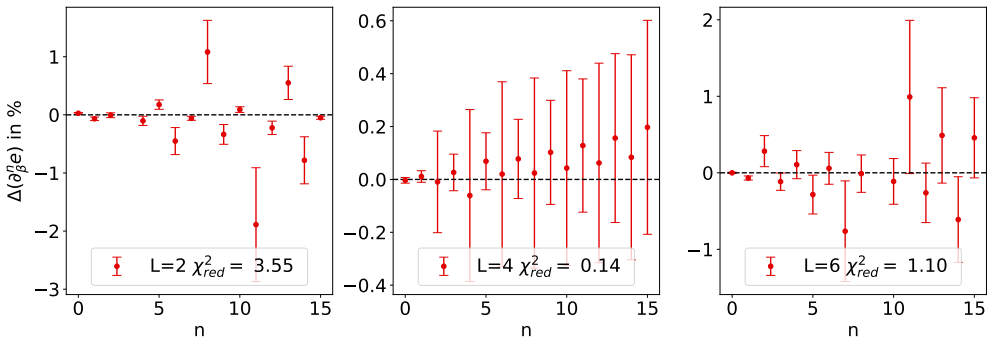


Figure A.3 $\beta = 0.375$

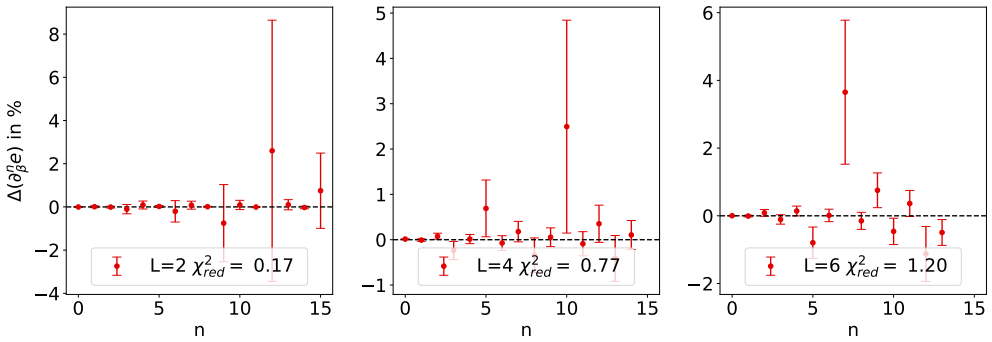


Figure A.4 $\beta = 0.425$

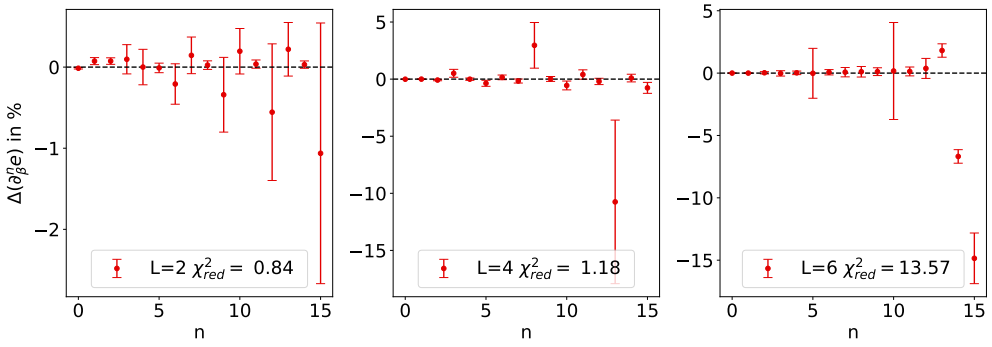
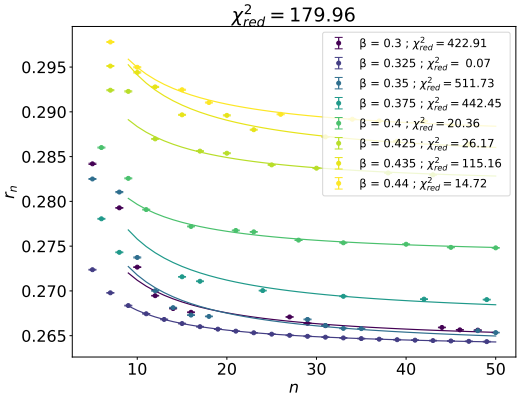


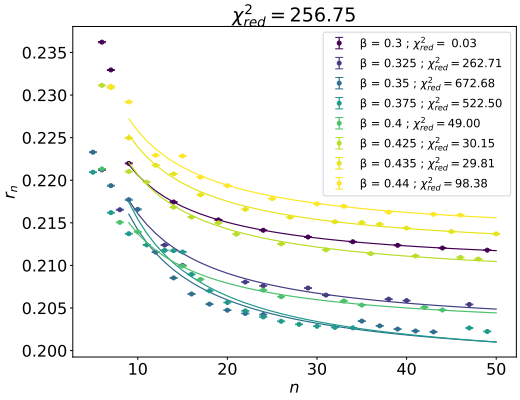
Figure A.5 $\beta = 0.435$

A.4 Radii of convergence

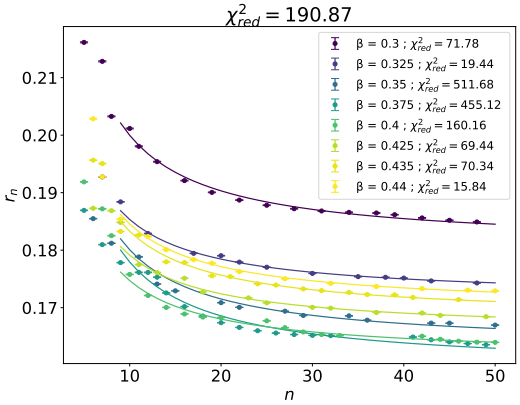
This section contains the plots for all the radii of convergence based on the lattice sizes of $L = 3$ up to 12, excluding $L = 6$ which can be seen in Figure 5.16.



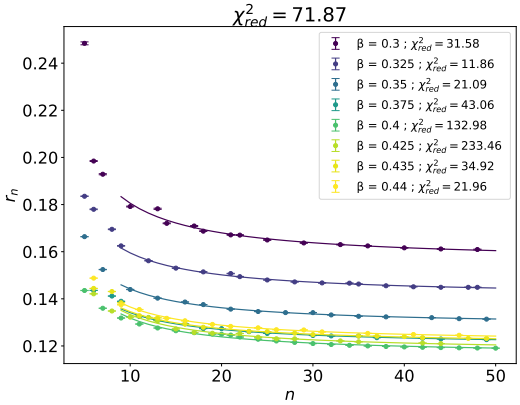
(a) $L = 3$



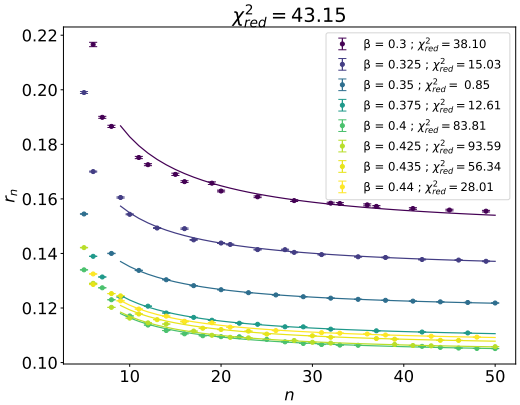
(b) $L = 4$



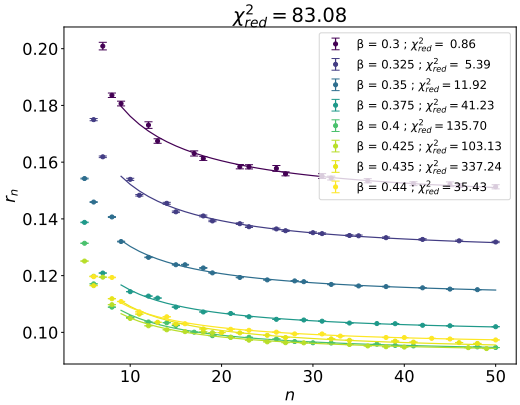
(c) $L = 5$



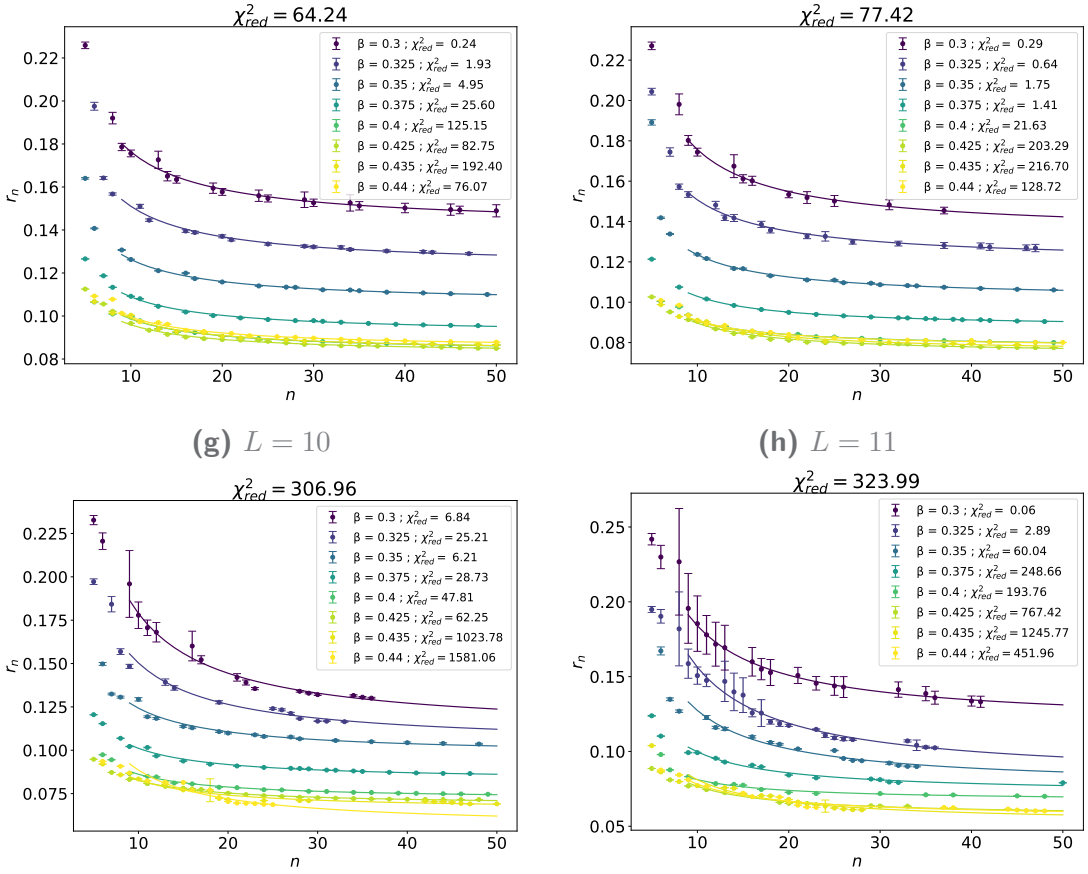
(d) $L = 7$



(e) $L = 8$



(f) $L = 9$



(g) $L = 10$ **(h)** $L = 11$

(i) $L = 12$ **(j)** $L = 13$

Figure A.6 Plots containing the fits of the radii of convergence for $L = 2$ up to $L = 13$. The underling data is based on the internal energy e

A.4.1 Explanation for omitting $L = 2$

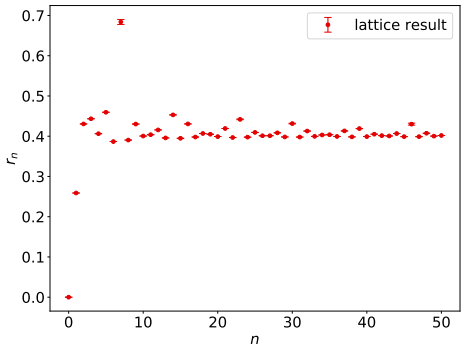
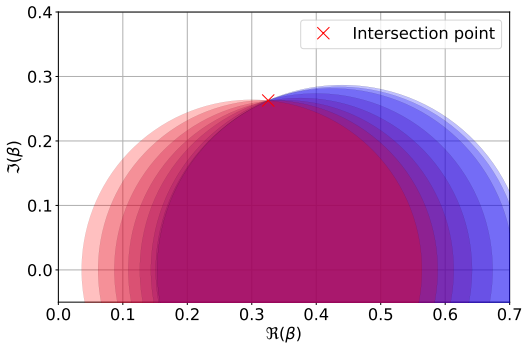


Figure A.7 Series elements from the root test for a lattice size of $L = 2$

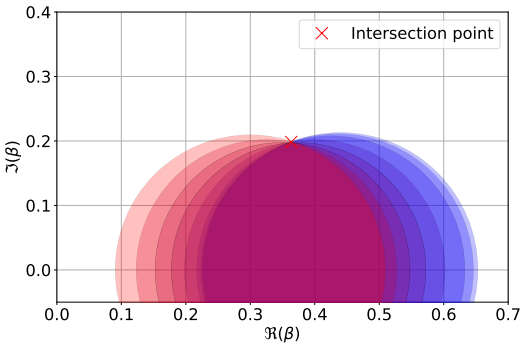
In the further analysis of the radii of convergence & the determination of the Fisher zeros, the case $L = 2$ will be left out due to the fact that, as Figure A.7 shows, no monotone series can be constructed. While the series itself seems correct and fits nicely into the pattern of the limit inferior, since it is incompatible with the method used to determine the limit inferior, $L = 2$ is omitted. However, in the systematic error section, the data points are re-included in those fitting models where there is no problem, i.e. those without the inferior approximation.

A.5 Determination of the Fisher Zeros

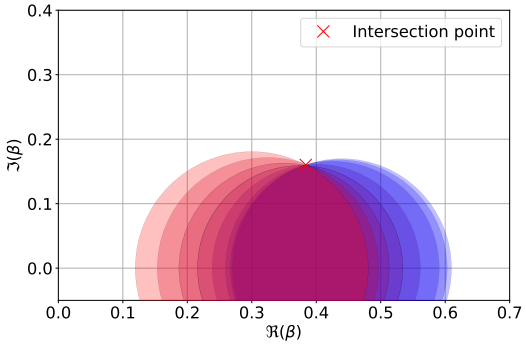
This section contains the plots used to determine the Fisher zeros from $L = 3$ to $L = 13$. The width of the black edges of the circle is equal to twice the statistical error calculated using the bootstrap method.



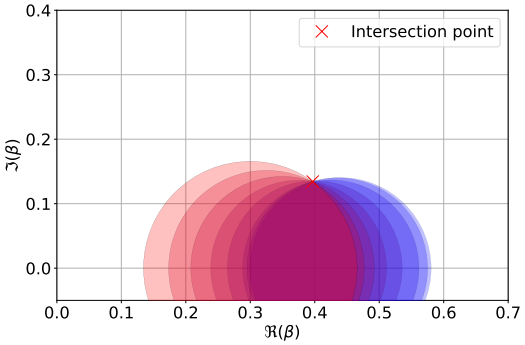
(a) $L = 3$



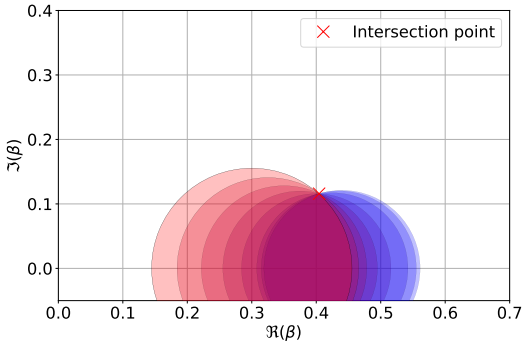
(b) $L = 4$



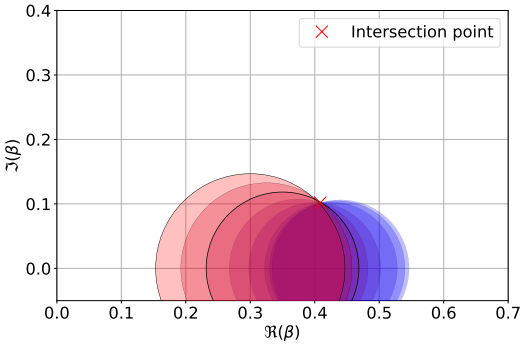
(c) $L = 5$



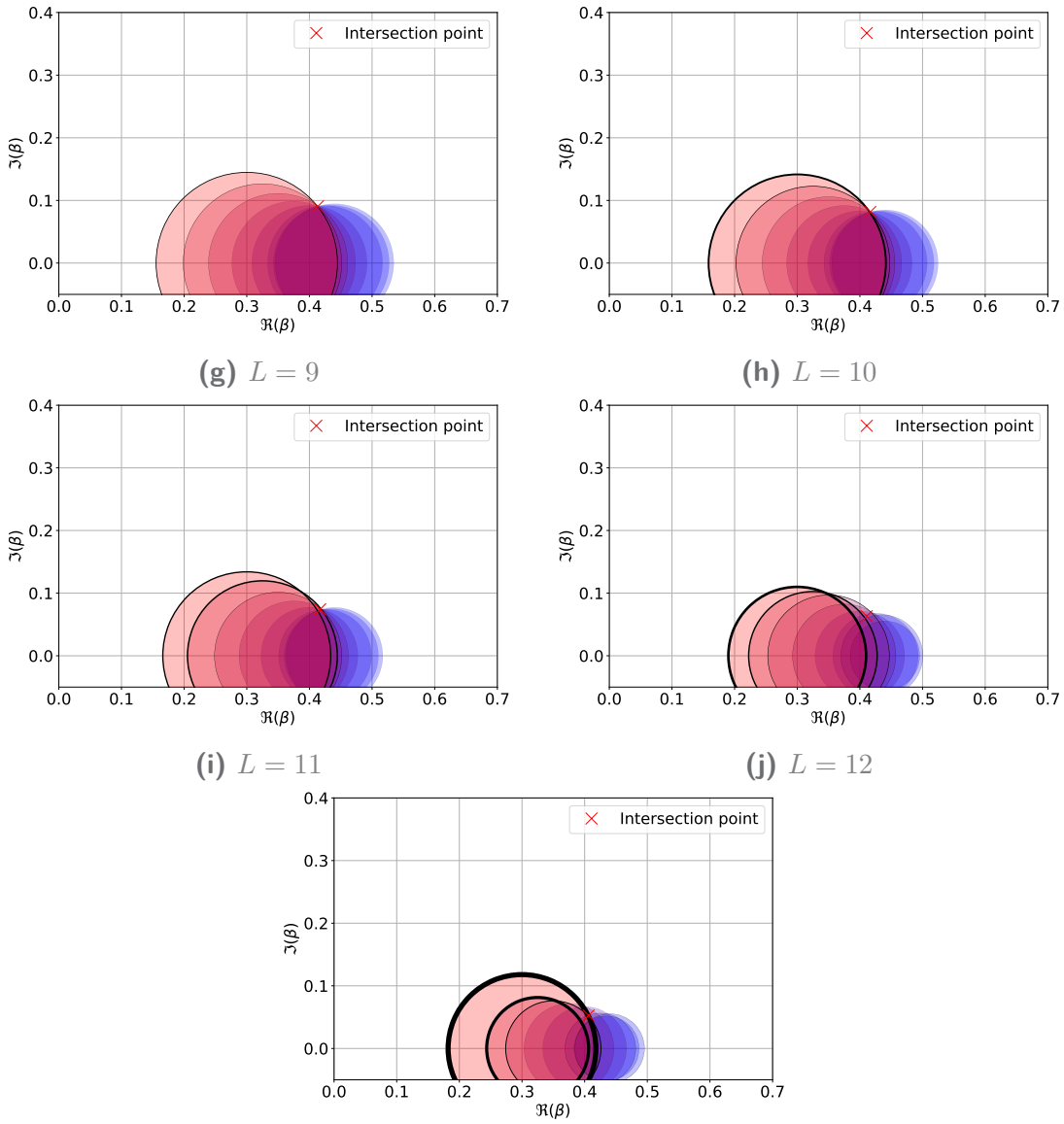
(d) $L = 6$



(e) $L = 7$



(f) $L = 8$



(g) $L = 9$ **(h)** $L = 10$

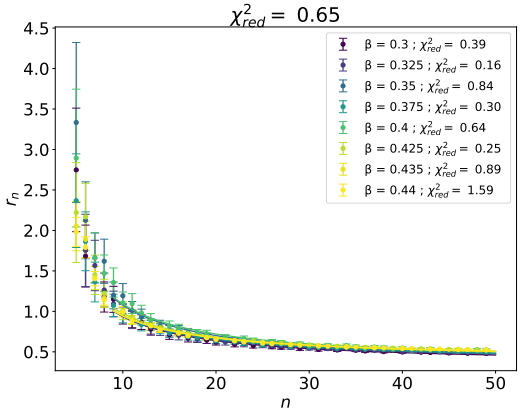
(i) $L = 11$ **(j)** $L = 12$

(k) $L = 13$

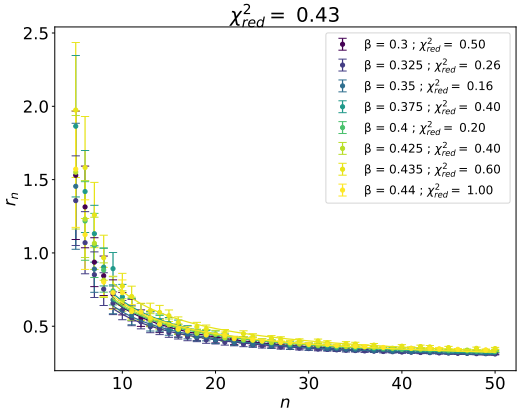
Figure A.8 Plots containing the fits of the interceptions points of the radii of convergence resulting in the Fisher zeros for $L = 3$ up to $L = 13$. The underling data is based on the internal energy e

A.6 Plots using the magnitude m as the observable

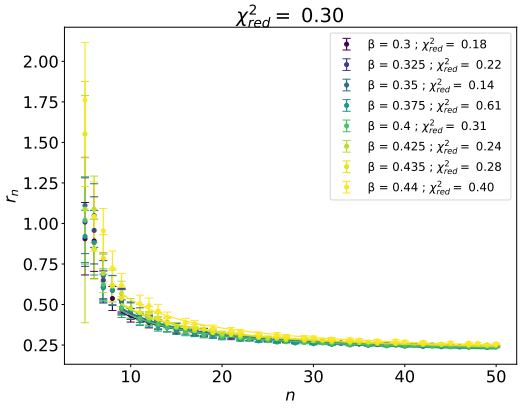
A.6.1 Plots of the radii of convergence



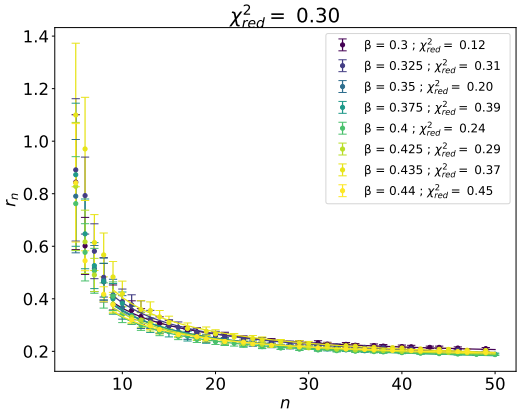
(a) $L = 2$



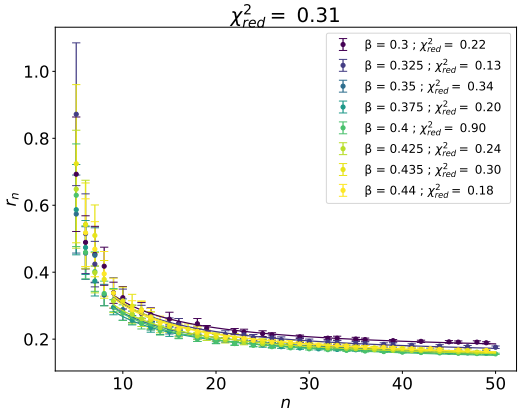
(b) $L = 3$



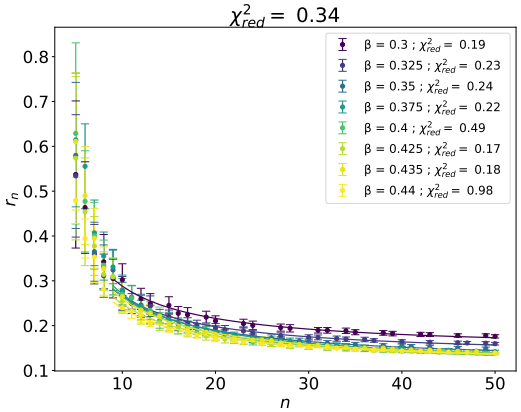
(c) $L = 4$



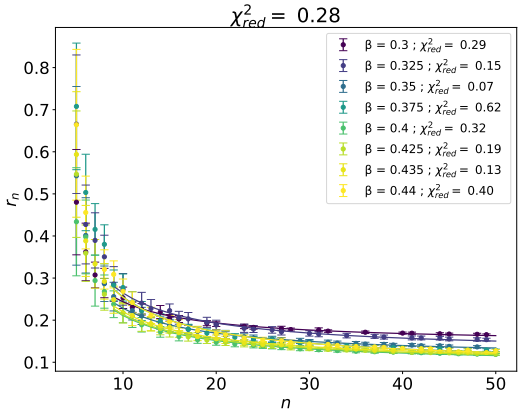
(d) $L = 5$



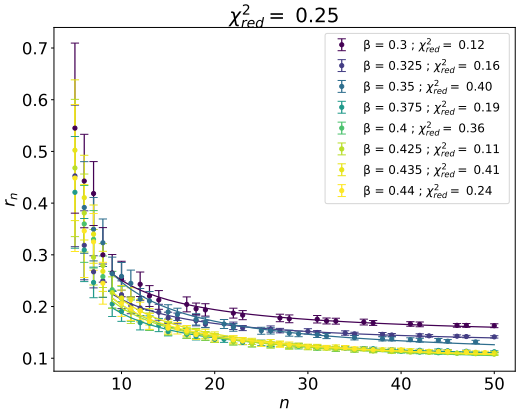
(e) $L = 6$



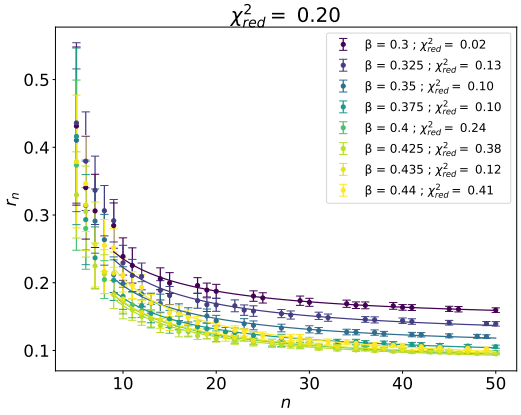
(f) $L = 7$



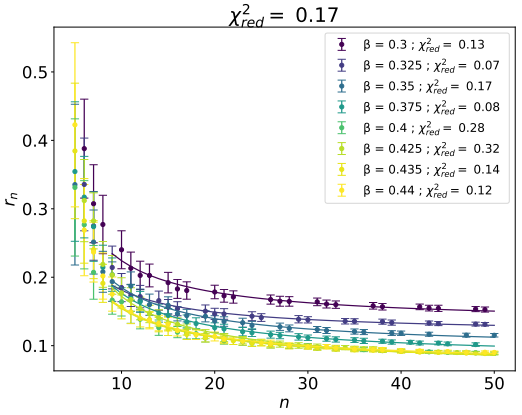
(g) $L = 8$



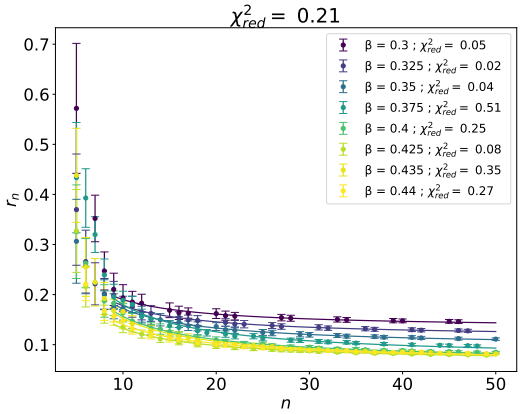
(h) $L = 9$



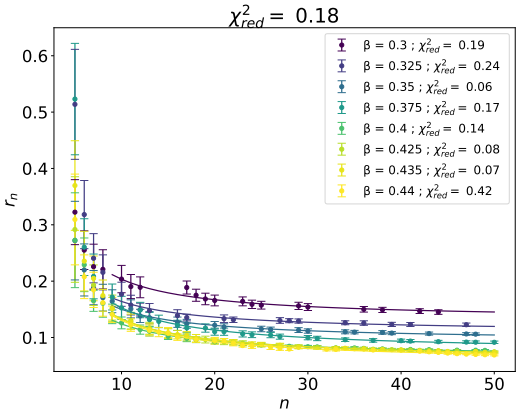
(i) $L = 10$



(j) $L = 11$



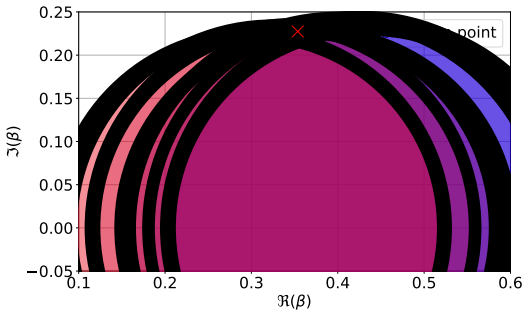
(k) $L = 12$



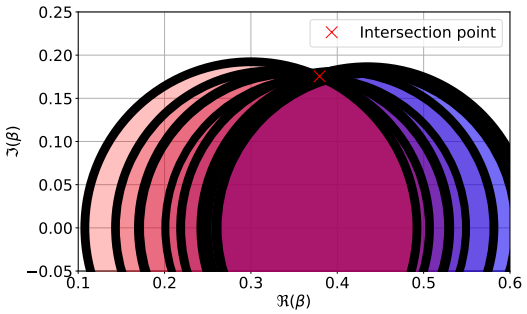
(l) $L = 13$

Figure A.9 Plots containing the fits of the radii of convergence for $L = 2$ up to $L = 13$. The underling data is based on the internal energy e

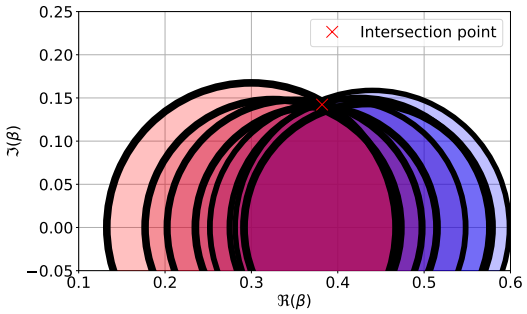
A.6.2 Plots of the Fisher zeros



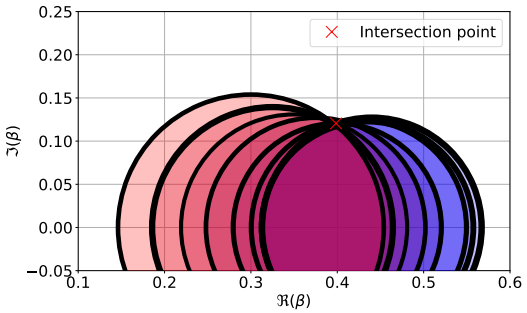
(a) $L = 3$



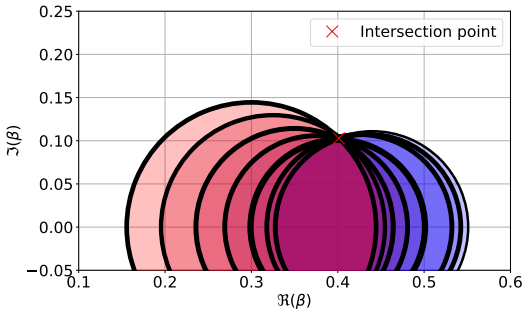
(b) $L = 4$



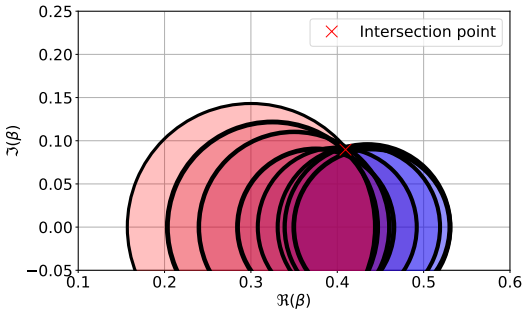
(c) $L = 5$



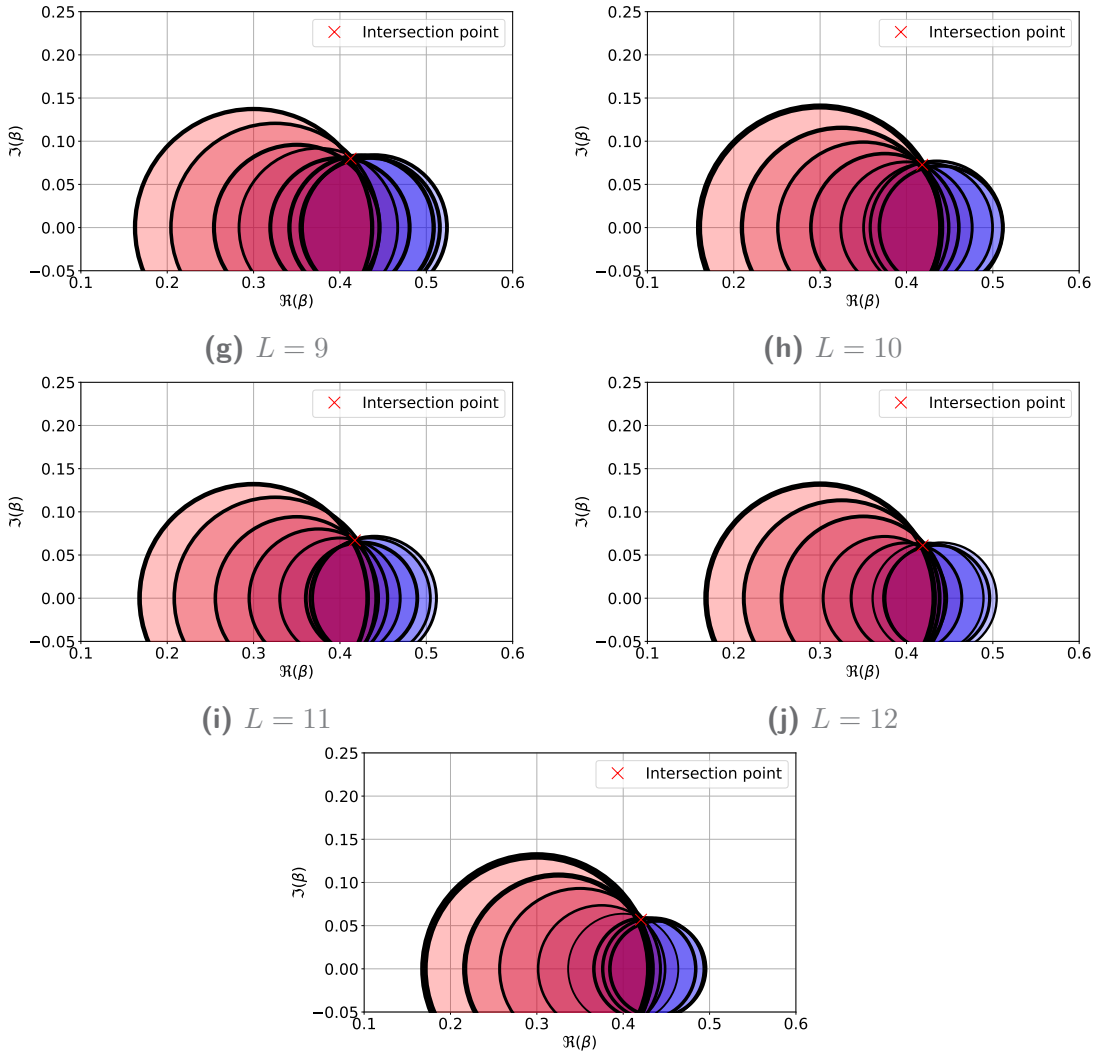
(d) $L = 6$



(e) $L = 7$



(f) $L = 8$



(g) $L = 9$ **(h)** $L = 10$

(i) $L = 11$ **(j)** $L = 12$

(k) $L = 13$

Figure A.10 Plots containing the fits of the interceptions points of the radii of convergence resulting in the Fisher zeros for $L = 2$ up to $L = 13$. The underling data is based on the internal energy m

A.7 Models used for the total error

Below is a list of all the unique functions used to determine the total error, where the letters a to e represent the possible fit parameters.

$$
\begin{aligned}
f(n) &= a + \frac{b}{n} \\
f(n) &= a + b\exp(-cn) \\
f(n) &= a + \frac{b}{n-c} \\
f(n) &= a + \frac{b}{n-c} + \frac{d}{(n-c)^2} \\
f(n) &= a \\
f(n) &= a + b\frac{\log(|d(n-c)|)}{d(n-c)} \\
f(n) &= a + \frac{b}{n-c} + \frac{d}{(n-e)^2} + \frac{e}{(n-c)^3} \\
f(n) &= a + \frac{b}{(n-c)^2} \\
f(n) &= a + \frac{b}{(n-c)^3} \\
f(n) &= a + \frac{b}{log(|c(n-d)|))}
\end{aligned}
\tag{A.12}
$$

Bibliography

[1] Bazavov, A. *et al.*, "The QCD Equation of State to $\mathcal{O}(\mu_B^6)$ from Lattice QCD," *Phys. Rev. D*, vol. 95, no. 5, p. 054 504, 2017.

[2] Bhattacharjee, S. M. and Khare, A., "Fifty years of the exact solution of the two-dimensional Ising model by Onsager," *Curr. Sci.*, vol. 69, pp. 816–820, 1995, [Erratum: Curr.Sci. 71, 493 (1996)].

[3] Borsanyi, S. *et al.*, "Leading hadronic contribution to the muon magnetic moment from lattice QCD," *Nature*, vol. 593, no. 7857, pp. 51–55, 2021.

[4] Königsberger, K., *Analysis 1* (Springer-Lehrbuch), German, ISBN: 978-3-540-40371-5.

[5] Malsagov, M. Y.; Karandashev, I. M., and Kryzhanovsky, B. V., *The analytical expressions for a finite-size 2d ising model*, 2017.

[6] Onsager, L., "Crystal statistics. 1. A Two-dimensional model with an order disorder transition," *Phys. Rev.*, vol. 65, pp. 117–149, 1944.

[7] Feynman, R. P. (P., *Statistical mechanics: a set of lectures by R. P. Feynman* (Frontiers in physics). 1972, Notes taken by R. Kikuchi and H. A. Feiveson. Edited by Jacob Shaham, ISBN: 0-8053-2508-5, 0-8053-2509-3 (paperback).

[8] Krauth, W., "Statistical mechanics: algorithms and computations," Jan. 2006.

[9] Ferdinand, A. E. and Fisher, M. E., "Bounded and Inhomogeneous Ising Models. 1. Specific-Heat Anomaly of a Finite Lattice," *Phys. Rev.*, vol. 185, pp. 832–846, 1969.

[10] Deger, A. and Flindt, C., "Determination of universal critical exponents using Lee-Yang theory," *Phys. Rev. Research.*, vol. 1, p. 023 004, 2019.

[11] Gattringer, C. and Lang, C. B., *Quantum chromodynamics on the lattice*. Berlin: Springer, 2010, vol. 788, ISBN: 978-3-642-01849-7, 978-3-642-01850-3.

[12] Wolff, U., "Collective Monte Carlo Updating for Spin Systems," *Phys. Rev. Lett.*, vol. 62, p. 361, 1989.

[13] Varnhorst, L., *Selected topics of lattice gauge theory*, Lecture notes, 2020.

[14] Efron, B., "Bootstrap Methods: Another Look at the Jackknife," *Annals Statist.*, vol. 7, no. 1, pp. 1–26, 1979.

[15] Fodor, Z.; Giordano, M.; Günther, J. N., *et al.*, "Trying to constrain the location of the QCD critical endpoint with lattice simulations," *Nucl. Phys. A*, vol. 982, Antinori, F.; Dainese, A.; Giubellino, P.; Greco, V.; Lombardo, M. P., and Scomparin, E., Eds., pp. 843–846, 2019.

[16] Haerter, J. O., *Statistical mechanics, the ising model and critical phenomena*, Lecture notes, 2017.

Name, Vorname:

Erklärung

gem. §15 Abs. 6 der Prüfungsordnung vom 25.11.2019

Hiermit erkläre ich, dass ich die Bachelor-Thesis selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

....
Datum Unterschrift

Erklärung

Hiermit erkläre ich mich damit einverstanden, dass meine Abschlussarbeit (Bachelor-Thesis) wissenschaftlich interessierten Personen oder Institutionen und im Rahmen von externen Qualitätssicherungsmaßnahmen des Studienganges zur Einsichtnahme zur Verfügung gestellt werden kann.
Korrektur- oder Bewertungshinweise in meiner Arbeit dürfen nicht zitiert werden.

....
Datum Unterschrift