

# Phase Transition Tunneling Time in a System of Hard Disks

Diploma Thesis  
in Theoretical Physics

presented by

Thomas Bisig  
*im Städtli 20, 8872 Weesen*

under the supervision of

Prof. M. Troyer  
*Institut für theoretische Physik, ETH Hönggerberg, 8093 Zürich*

September 6, 2006



# *Abstract*

We determine the dependence of the tunneling time on the size of a system of hard disks through a first-order phase transition using the Wang-Landau algorithm. The algorithm overcomes the critical slowing down in the vicinity of a first-order phase transition observed for conventional Monte Carlo simulations. We show that the tunneling times obey a linear law which shall be determined. We additionally introduce an order parameter which measures the systems orientational order. This order parameter allows the disks to make preferred moves and could make the tunneling through the phase transition region even faster by forcing the disks to move into a more ordered state.



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Contents</b>	<b>v</b>
<b>List of Figures</b>	<b>vii</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 System of hard disks</b>	<b>3</b>
2.1 Description of the system . . . . .	3
2.2 Thermodynamics of the hard disk system . . . . .	4
<b>3 Simulation</b>	<b>7</b>
3.1 Wang-Landau algorithm . . . . .	7
3.1.1 Theory . . . . .	7
3.1.2 The algorithm . . . . .	8
3.1.3 Application to the problem . . . . .	10
3.2 Simulation details . . . . .	11
3.2.1 Density updates . . . . .	11
3.2.2 Disk moves . . . . .	11
3.2.3 Hardware and Software/Implementation . . . . .	12
3.2.4 Modified update . . . . .	13
3.2.5 Chosen parameters . . . . .	13
3.2.6 Mixing the system . . . . .	13
3.2.7 Overlapping check . . . . .	13
3.2.8 Scripts and additional code . . . . .	14
3.3 Results . . . . .	14
3.3.1 Density of states . . . . .	14
<b>4 Tunneling times</b>	<b>17</b>
<b>5 Order parameter</b>	<b>21</b>
5.1 Implementation . . . . .	22
5.2 Results . . . . .	23
<b>6 Conclusion and Outlook</b>	<b>25</b>
<b>A Visualize disk system</b>	<b>27</b>

**Bibliography**

**29**

# List of Figures

2.1	Low and high density state of a system of hard disks. . . . .	4
3.1	Evolution of the flatness parameter during a simulation. . . . .	10
3.2	Density of states for a system of nine disks. . . . .	15
4.1	Tunneling times through the first-order phase transition from the liquid to the solid phase. . . . .	18
4.2	Tunneling times through the first order phase transition. . . . .	18
5.1	Grid cell method and order parameter. . . . .	22
5.2	Density of states in order parameter direction for 16 disks. . . . .	24



# Chapter 1

## Introduction

The two dimensional hard disk fluid has been very important in the development of the theories of liquids and liquid structures. Its solid-liquid melting transition however has been discussed extensively and controversially in the last three decades and it is still an unsolved problem [1]. The two dimensional case differs from its three dimensional analogous because the two dimensional solid possesses only a quasi long range positional order, while the three dimensional solid is truly long range ordered.

There exist several theoretical approaches for the description of melting in two dimensions. One approach follows the ideas of Kosterlitz and Thouless [2], where two order parameters are introduced but related to different topological defects. Alternative theories are the one of Chui [3] or the one proposed by Glaser and Clark [4], which both predict a first-order phase transition between liquid and the solid phase.

Not only theoretical but also numerical investigations of the melting transition can be done in several ways. Numerical simulations have become an important part in statistical physics. In particular, the so called Monte Carlo (MC) simulations have proven to be very helpful in obtaining thermodynamic properties, investigating phase transitions and critical phenomena. However, conventional MC simulations are under certain circumstances not applicable to real physical problems. The critical slowing down of MC simulations in the vicinity of a first-order phase transition is an example of such a physical problem. Thus many modifications have been proposed over the last years to improve the algorithm.

One of these modifications to the conventional MC algorithm is the one proposed by Wang and Landau [5, 6], which elegantly calculates the density of states in computer simulations directly. The density of states  $\Omega(E)$  is directly related to the entropy of a system through  $S = \ln \Omega(E)$ , where the Boltzmann constant is omitted and we have assumed that the density of states is in the microcanonical ensemble, that is it only depends on the energy  $E$ . By knowing the density of states, we can calculate all thermodynamic properties of interest through the entropy. The Wang-Landau algorithm assumes an *a priori* density of states function which is successively approximated during the simulation such that the density of states function converges to its true value. The algorithm's

distinguishing feature is the dynamic update of its acceptance rule: the density of states is modified at every simulation step rather than between runs.

First-order phase transitions, as in the problem at hand, require more sophisticated methods than conventional MC simulation as cluster flipping algorithms [7,8] or simple Metropolis importance sampling [9]. The main differences between phase transitions of first order and of higher order are that in the first kind of phase transitions no divergent quantity can be exhibited and that the two phases coexist in the region of the phase transition. This means that in a solid-liquid transition both the solid and the liquid phase exist in some region of configuration space. The Wang-Landau MC algorithm has proven to be the better choice when it comes to physical problems in which one has to overcome a phase transition of first order. The main reason for this is the equal visit probability for every state, where we have to discretise our continuous system in order to be able to apply the algorithm.

The system's states are given by the different configurations and distributions of the disks over the so-called box which has a specific width and length depending on the number of disks under surveillance. The disks can freely be moved but are not allowed to overlap, which physically can be achieved by defining an appropriate potential function. The disk's radius is directly coupled to the system's density. This leads us to changing radii to vary the density, which will be necessary for the Wang-Landau algorithm.

We also introduce an order parameter which forces the disks to prefer certain moves, namely it moves the disk preferably into states with higher order and therefor the system is much more likely to be in a state of high density. The order parameter is a sort of an entropy driven constraint on the disk moves.

In this report we are interested in the tunneling times  $\tau(N)$  from low density to high density states through the phase transition using the Wang-Landau MC algorithm. We investigate the dependence of the tunneling time  $\tau(N)$  on the system size  $N$ , that is the number of hard disks. We show that the tunneling times obey a linear law.

We present in Chapter 2 the system consisting of hard disks and its thermodynamic properties. In Chapter 3 we first start with explaining the Wang-Landau algorithm we used and its application to the problem as well as some results obtained by applying the algorithm to the problem. The simulation details can be found therein as well. In Chapter 4 we present the results for the tunneling times and its conclusion. The order parameter and its implementation are introduced in Chapter 5. The conclusion and outlook are presented in Chapter 6.

## Chapter 2

# System of hard disks

### 2.1 Description of the system

We consider the two dimensional case of a box consisting of  $N$  hard disks. We restrict the choice of  $N$  to square numbers  $N = n^2$ ,  $n \in \mathbb{N}$  as this makes the simulation easier and does not put any constraints on the results. The hard disks can be in any configuration that takes the non-overlapping of any two disks into account. Mathematically spoken, the potential energy of the system is given by

$$U_{ij} = \begin{cases} \infty & \text{if } r_{ij} < r_D \\ 0 & \text{if } r_{ij} > r_D \end{cases},$$

where  $r_{ij}$  is the distance between disk  $i$  and  $j$  and  $r_D$  is the disk radius. During the simulation we have to change the disk radius for all disks simultaneously. We are interested in the transition time through the phase transition from a liquid to a solid state (see Fig. 2.1). We call a state solid if the hard disk system is in the configuration of closest packing and has maximal density<sup>1</sup>. The state of closest packing fills a rectangle with ratio of width to length of  $2/\sqrt{3}$ , where we choose the width of the box to be the longer side and for simplicity to be  $L$ . The volume of the box is given by

$$A = \sqrt{3}/2L^2 = 2\sqrt{3}n^2r_{\text{initial}}^2 = 2\sqrt{3}Nr_{\text{initial}}^2,$$

where  $N = n^2$ ,  $n \in \mathbb{N}$  and  $r_{\text{initial}}$  is the initial radius of the disks, which we choose  $r_{\text{initial}} = \frac{1}{2}$ . The absolute value stays the same throughout the simulation. To change the density we alter the radius of the disks (see 3.2.1). With that, the reduced density  $\rho$  is defined as

$$\rho \equiv \frac{N\sigma^2}{A} = \frac{2(2r_D)^2}{\sqrt{3}},$$

where  $\sigma = 2r_D$  denotes the diameter of the disks. As the volume of the box is fixed, the radius of the disks have to be in the interval  $r_D \in [0, \frac{1}{2}]$ , otherwise the disks wouldn't fit into the box. The solid state (and therefor closest packing) corresponds to a maximal density  $\rho_{\text{max}} = \frac{2}{\sqrt{3}} \approx 1.15$ . The liquid state

---

<sup>1</sup>Actually a state of maximal density implies the state of closest packing.

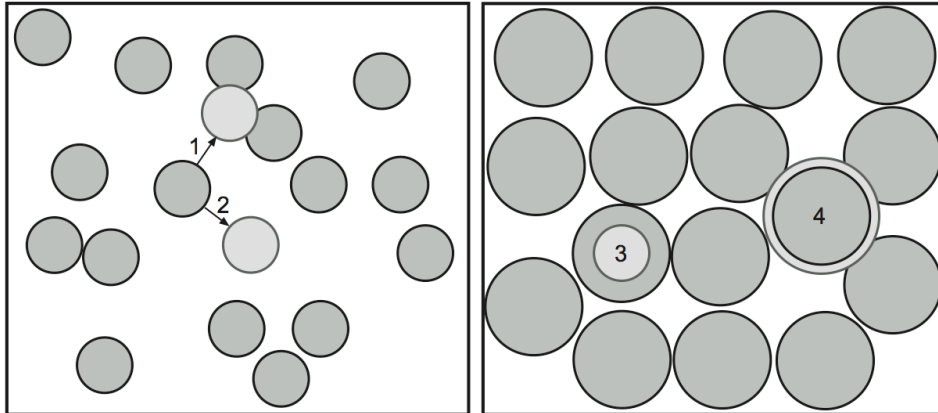


Figure 2.1: The low density state  $\rho \approx 0.27$  on the left tunnels through a first order phase transition to a high density state  $\rho \approx 0.82$  on the right. The box has periodic boundary conditions in both directions. While disk displacement (1) is not allowed because the disks would overlap, the displacement (2) is accepted. After moving a disk we change the radius of all disks. While radius change (3) is allowed for all the disks, the change (4) is not allowed because two or more disks would overlap after the update. Actually neither of the figures show the periodic boundary conditions as one might expect. One can think of it as a special case of a configuration where none of the disks reach 'out' of the boundaries of the box.

corresponds to the minimal density, which trivially is  $\rho_{\min} = 0$ .

Restricting the number of disks to square numbers allows us to use a simplified box instead of a general one. For a general number of disks we would have to construct a non-rectangular box, but as we are interested in the general behavior of the system, depending on its size, it doesn't matter what number of disks we are investigating.

We choose the boundary conditions to be periodic in both x- and y-direction to get rid of boundary effects completely. The box therefor forms a torus and the disks move on the surface of the torus. We have to keep in mind that although boundary effects vanish by choosing periodic boundary conditions, finite size effects still remain, since all lengths in the problem are periodic with the system size as period.

## 2.2 Thermodynamics of the hard disk system

As the description of the model lies in thermodynamics, we are interested in thermodynamic quantities and how we can derive them. Once calculated, the partition function enables us to obtain the equation of state for the system of hard disks. The partition function in terms of the number of disks  $N$ , the volume  $V$  and the temperature (the Boltzmann constant is neglected)  $T = 1/\beta$  is given by

$$Z(N, V, \beta) = \sum_{\mathcal{C}} e^{-\beta \sum U_{ij}},$$

where the first sum goes over all possible configurations  $\mathcal{C}$ , the second sum in the exponent over pairs of disks in the system and  $U_{ij}$  is the potential energy of

disk  $i$  with disk  $j$ . Due to our simple potential for the interaction of two disks we can rewrite the exponential term to

$$e^{-\beta \sum U_{ij}} = \begin{cases} 0 & \text{if } r_{ij} < r_D \\ 1 & \text{if } r_{ij} > r_D \end{cases}$$

and therefor the partition function simplifies to

$$Z(N, V, \beta) = \Omega(N, V),$$

where  $\Omega(N, V)$  is the number of possible states for a given  $N$  and  $V$ . The temperature independent partition function emerges because of the special form of the potential function and is given by the density of states directly. Thus it is not necessary to simulate multiple times for different temperatures, which has to be done for conventional MC simulations. Introducing the entropy  $S(N, V) = \ln \Omega(N, V)$  leads to the free energy  $F = -T \ln Z = -TS$ , which allows us to write the equation of state as

$$P = -\frac{\partial F}{\partial V} = T \frac{\partial}{\partial V} \ln \Omega(N, V).$$

It has to be noted, that for our duty we have no need for the density of states  $\Omega$  in dependence of the volume  $V$ , but rather of the density  $\rho$ . Therefor, we cannot apply the equation of state in the form above but in a different form regarding the density to be the parameter of choice.



## Chapter 3

# Simulation

### 3.1 Wang-Landau algorithm

As Wang and Landau state in their published papers [5,6], the proposed algorithm to obtain the density of states is not restricted to energy space, but can be used to calculate the density of states for any parameter of interest by a random walk in the corresponding space. They showed that the modified MC simulation overcomes the critical slowing down at a first-order phase transition and, moreover, is especially useful to find the density of states for complex systems with a rough landscape since all possible parameter levels are visited with the same probability. In this chapter, we first introduce the theory of the Wang-Landau algorithm and then apply it to the system under investigation (see Chapter 2) and show some first simulation results. There are several things worth mentioning we had to deal with while setting up the simulation. These can be found in Section 3.2.

#### 3.1.1 Theory

The ability to overcome the critical slowing done in the vicinity of a first-order phase transition has already been mentioned as a problem solved by the proposed algorithm. Furthermore, it's also especially useful as we obtain the density of states directly and therefore are able to estimate thermodynamic quantities such as internal energy and specific heat capacity by calculating (micro)canonical averages at any temperature. Therefore, we avoid multiple simulations at different temperatures, but are able to calculate the free energy and entropy. These quantities can not be obtained directly by conventional MC simulations.

The root of the problem of conventional MC simulation is that the probability of visiting a state is not equal for all states. So a very rare state is very unlikely to be visited what results in problems in the vicinity of phase transitions. The Wang-Landau algorithm resolves this problem elegantly and starts with an initial guess for the density of states and then approximates the real density of states at each simulation step (and not only after each simulation).

Let's assume we have a conventional MC simulation with the energy  $E$  as parameter. Then the probability for visiting a state with energy  $E$  is e.g. given

by

$$P(E, T) = \Omega(E)p(E, T) = \Omega(E)e^{-E/T},$$

where we omit the Boltzmann constant and  $\Omega(E)$  is the number of states with energy  $E$ . For a first-order phase transition there exist two phases with different energies at the critical temperature. As we want to simulate from one phase to the other we have to continuously change the energy and, therefore, have to go through a minimum of states (between the two phases). As the probability between the two peaks decreases exponentially with the system size, we have to wait a long time until we tunnel through this minimum or find a better algorithm. The Wang-Landau is such an algorithm that suits problems of that kind much better.

Rather than choosing the probability to visit a given state with energy  $E$  to be  $\propto e^{-E/T}$ , we set the probability proportional to the reciprocal of the density of states  $\propto 1/\Omega(E)$ . The probability for visiting any state of energy  $E$  is then given by

$$P(E, T) = \Omega(E)p(E, T) = \Omega(E)\frac{C}{\Omega(E)} = C,$$

where  $C$  is the proportional constant. The goal of a partial run of this algorithm is to generate a flat histogram for the energy distribution. This histogram counts the number of times the algorithm has visited a state with energy  $E$ . This is done by modifying the (initially) estimated density of states to reproduce a flat histogram over the energy space and simultaneously making the density of states converge to the true value. We eventually update the density of states at each simulation step of the random walk and use the updated density of states to perform the next simulation step. The modification factor has to be chosen carefully and should be very close to one at the end of the simulation as this assures, that the density of states is near its true value.

### 3.1.2 The algorithm

The following is the Wang-Landau algorithm in pseudo code, remarks and notes follow afterwards.

- (1) Initialize density of states with a guess
- (2) While the modification factor is not near one do
- (3)     Reset histogram
- (4)     While histogram is not flat do
- (5)         Pick a random state/site/disk
- (6)         Make a local Metropolis update
- (7)         Increase the histogram at the current energy
- (8)         Increase the estimate for the density of states
- (9)     Reduce modification factor
- (10) End

As the density of states is *a priori* unknown, we start by guessing a (probably very bad) initial density of states  $\Omega(\rho)$  (1), which very commonly is chosen to be equal to  $\exp(1)$  for all states. The investigated parameter space shall be

denoted by  $\rho$ <sup>1</sup>. For a continuous system we have to divide the investigated parameter range into bins of a specific size. This has to be done in the current case (see Chapter 3.2). The main loop (2) ensures that the modification factor  $m$  converges to one the longer the simulation runs (9). Thus the modification of the density of states in every Metropolis update is finer in every loop. The modification factor is reduced by any function satisfying the convergency to one. We choose a square root function, thus the update is as follows:

$$m \leftarrow \sqrt{m}$$

We stop the simulation (10), if the modification factor is smaller than a pre-defined value  $m_{\text{final}}$ , which can be  $m_{\text{final}} = \exp(10^{-z}) \approx 1$  for  $z = 3, \dots, 8$ . The number of loops, the number of times of resetting the histogram  $H(\rho)$  is  $\lceil z \frac{\ln 10}{\ln 2} \rceil^2$ , if we start with a modification factor  $m_{\text{start}} = \exp(1)$ . For  $z = 8$  we have to perform 27 loops. Every time the modification factor  $m$  has been reduced, we have to reset the histogram  $H(\rho)$  (3). The histogram is a summary of how many times we have visited which state. As for the density of states, we have to divide the histogram into bins for a continuous systems. All this bins contain a number that represents the number of times the algorithm has visited that state. The inner loop (4) runs until the histogram is flat. Perfect flatness would mean that every state has been visited the same number of times, e.g. the bins of the histogram contain all the same number. As this is hardly possible, we construct criterions of flatness. Usually, the histogram has to satisfy, that the minimum number in a bin is bigger than  $x$  times,  $x < 1$ , the average number of visits of all bins:

$$H_{\min}(\rho) > x \bar{H}(\rho).$$

For small systems,  $x$  can be as high as 0.95, for bigger systems it is possible that such flatness can never be achieved. We have chosen  $x$  to be in the interval  $[0.7, 0.9]$ . An example of the evolution of the flatness parameter during a simulation can be seen in Fig. 3.1. Another difficulty arises if we want to perform a simulation in two parameter spaces; the flatness criterion has to be modified accordingly (see Chapter 5).

While the histogram is not flat, we pick a random state (5) and perform a local metropolis update (6). As the probability at a given parameter level is  $1/\Omega(\rho)$ , the probability of changing from the initial state  $\rho_1$  to the next state  $\rho_2$  is given by

$$p(\rho_1 \rightarrow \rho_2) = \min\left(\frac{\Omega(\rho_1)}{\Omega(\rho_2)}, 1\right).$$

Thus, if the density of states function is bigger for the state we are in than in the state we want to change to, we update with a probability  $p(\rho_1 \rightarrow \rho_2) = 1$ ; we certainly change to the new configuration. If the current density of states function is smaller than the new one, we update just with probability  $p(\rho_1 \rightarrow \rho_2) = \frac{\Omega(\rho_1)}{\Omega(\rho_2)}$ <sup>3</sup>.

<sup>1</sup>We will make a Wang-Landau MC simulation for the density, which we already have called  $\rho$ .

<sup>2</sup>The brackets  $\lceil \dots \rceil$  mean the next higher integer number.

<sup>3</sup>This can be done by drawing a random number  $\varphi \in [0, 1]$  and change from  $\rho_1$  to  $\rho_2$  if  $\varphi < p(\rho_1 \rightarrow \rho_2)$ .

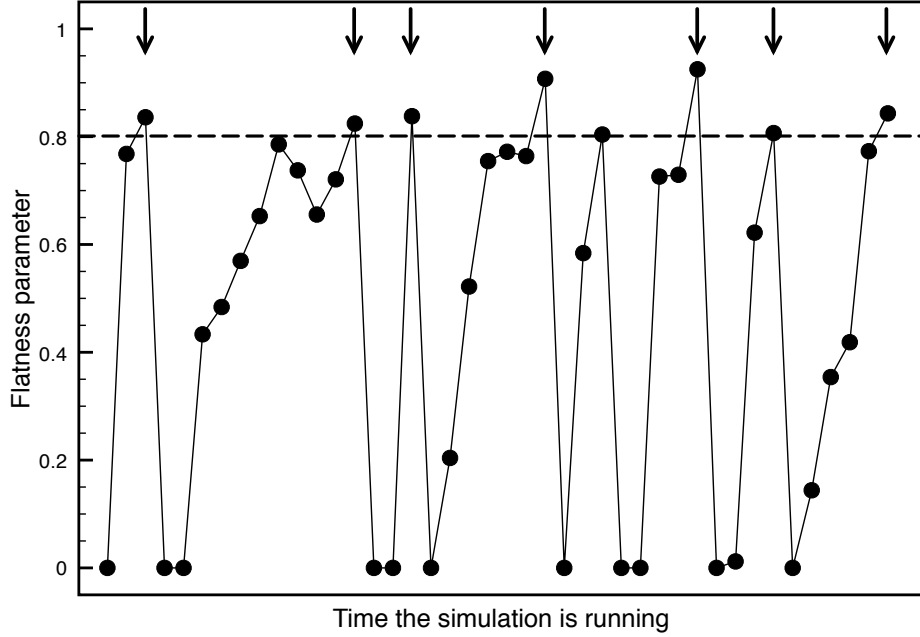


Figure 3.1: The evolution of the flatness parameter, which is checked for flatness every  $N_{\text{Sweeps}}$  during the simulation. This plot corresponds to the simulation of 16 disks with  $x = 0.8$  (the horizontal line indicates the flatness condition). The dots describe a check of the histogram on flatness. The arrows  $\downarrow$  point to a check at which the histogram fulfills flatness, thus  $H_{\min}(\rho) > 0.8\bar{H}(\rho)$  and the histogram is reset to 0.

This means that the more a state has already been visited, the less likely it is that it is visited again<sup>4</sup>. Every time we visit a state  $\rho$  we update the histogram (7)

$$H(\rho) \leftarrow H(\rho) + 1$$

to be able to perform the flatness calculation after  $N_{\text{Sweeps}}$  steps and modify the density of state function (8) for the updated state as well:

$$\Omega(\rho) \leftarrow \Omega(\rho) \cdot m.$$

No change in state means we are still in the same state, the modifications are made nevertheless. The further the simulation is<sup>5</sup>, the lesser the density of states  $\Omega(\rho)$  is modified. The modification factor has to converge to one, so finally we multiply the current density of states with a factor only slightly different from one, thus letting  $\Omega(\rho)$  almost unchanged.

### 3.1.3 Application to the problem

We now have to apply the proposed algorithm to the investigated disk system. We want to tunnel from the lowest density (liquid) to high density (solid), proposing the density to be the parameter of choice. Therefore a Metropolis

<sup>4</sup>This depends on what state we are currently.

<sup>5</sup>This is in the meaning of the more times we have reduced the modification factor  $m$ .

update consists of moving a particle and changing its radius to lead to a new density with respective probability. As we are only interested in the tunneling times, we can neglect some aspects of the Wang-Landau algorithm. We could have measured the time the algorithm needs starting from lowest density to reach the highest density, which means the highest bin of the density parameter. What we actually did, is measuring the time the algorithm needs to reach almost the highest density. As flatness can't be achieved without the highest investigated state being visited at least once, the histogram check in the discussed manner can be left. Instead, we have to check after some number of steps whether the state has already been visited and if yes, stop the algorithm and save the number of needed steps; if no, continue until it's the next check's turn.

## 3.2 Simulation details

### 3.2.1 Density updates

The density  $\rho$  of a system of hard disks is proportional to  $\propto r_D^2/A$ . Thus, there are two ways of changing the density, either by changing the volume of the box or the radii of all disks in the system. While changing the volume of the box involves recalculating every disk's neighbor list and position, changing the radii of all disks lets the system, especially the list of neighbors, unaltered (see 3.2.7). As density changes are driven by radius changes, we have to update the radius in every single step of the MC simulation. Radius updates can only be performed in the range  $r_D \in (0, r_{\max}]$ , making it necessary to check if any two disks overlap. For our simulations we took  $r_{\max} = 0.5$ .

The way, the radius update is performed is important too. A simple update by drawing a random number  $\omega \in (0, r_{\max}]$  and setting this as the new radius (with respective probability) doesn't simulate correctly. This means, that if we use this idea, the algorithm never tunnels through the phase transition. What we have to do is to define a maximal radius change  $\Delta$  and change the radius in the interval

$$r_D \leftarrow \omega \in [r_D - \Delta, r_D + \Delta].$$

We have chosen  $\Delta = 0.2$ . In Table 3.1 the ratio of successful disk radius changes can be seen. Roughly, every second trial was successful.

### 3.2.2 Disk moves

As the radius updates, the disk moves belong to a Metropolis step as well. The more we move disks (without any constraints or preferred moves), the more unordered the system becomes and therefor the disks generally are not in a state of high density. We start the simulation in the perfect crystal state (closest packing) and perform  $N_{\text{init}}$  configuration changes to lead the system into an arbitrary state<sup>6</sup> (see 3.2.6). In our simulations, we took  $N_{\text{init}} = 10^4$ . We made one disk displacement per radius change, but its possible to make more

<sup>6</sup>Actually the initialization process is done by the application *Initializer* (see Appendix).

No. of Disks	Displacement ratio	Density ratio
9	0.91910	0.52640
16	0.81960	0.48610
25	0.78030	0.44610
36	0.55460	0.46760
49	0.52500	0.46520
64	0.65530	0.48170

Table 3.1: This table summarizes the displacement and density change ratios. The ratios indicate the mean probability for the success of an update process (movement or radius) as not every step is allowed, due to probable overlapping of two (or more) disks.

moves per density update. As for the radius changes, we define a maximal allowed displacement  $\delta$ . We have chosen  $\delta = 0.05$ . The respective disk move ratios can be seen in Table 3.1. The larger the investigated system gets, the lesser likely it is for a single move to be accepted. The maximal allowed disk displacement  $\delta$  should not be chosen to depend on the disk radius, but be held constant throughout the whole simulation.

### 3.2.3 Hardware and Software/Implementation

Simulations were done on the hreidar cluster [10] of the ETH Zurich, making it possible to run several simulations simultaneously. Although we ran our simulation only on one node<sup>7</sup>, getting the simulation to work was a time consuming task. The calculation time on hreidar is limited to 480 minutes, making it necessary to save the simulation state after some shorter time and starting a new simulation with the saved simulation state. We have written a shell script to start new simulations if the former one has finished (See Section 3.2.8)<sup>8</sup>. The simulation has been done in two steps. The first steps was to initialize the system, getting in a state of lowest density and saving this state to a file. The second step was the simulation itself, reading the saved file and starting the simulation. This second step is repeated until once (or several times) the highest investigated density has been visited.

The code has been written in *C++* and was initially tested on a Powerbook running *MacOS 10.4.7* using *Xcode* and its debugger to build and test the used applications. The boost library [11] has been used for various duties as for timing (*boost/timer.hpp*) or for generating random numbers (*boost/random.hpp*). To visualize the system (to check whether moving disks and checking overlapping criterion works) we used the simple but efficient *CImg* [12] library (see Appendix A).

To draw random numbers we use a lagged fibonacci (lagged\_fibonacci19937) from the boost library.

<sup>7</sup>Simulations would be much faster using MPI but because of the limited time range this is left for future investigations, probably for larger systems.

<sup>8</sup>In order to conserve the login and job submitting server, we have tried to submit a shell script itself, calling the new job automatically after one has finished. For simplicity we finally took the former method as new problems arose that the time left didn't allow to solve.

### 3.2.4 Modified update

Instead of updating the density of states in the discussed way, we use an update procedure as follows:

$$\ln \Omega(\rho) \leftarrow \ln \Omega(\rho) + \ln m,$$

in order to fit all possible  $\Omega(\rho)$  into double precision numbers for the system under investigation. Of course this modified update method has to be taken into account for the calculation of the probabilities, as not the real density of states are saved, but its logarithm.

### 3.2.5 Chosen parameters

At the beginning of our study of the problem we used an initial value for the modification factor of  $m_{\text{initial}} = \exp(1) \approx 2.71828$ , but during the optimization of the implementation it emerged that it's much better to use at least  $m_{\text{initial}} = \exp(2)$ , which we used throughout our simulations. However, a too large choice of the initial value of  $m_{\text{initial}}$  would lead to large statistical errors. The choice of the bin size for the parameter under investigation is also very important: (1) A large bin size means the values of the density of states  $\Omega(\rho)$  in each bin is very large, which makes it impossible to exponentiate these numbers; (2) The smaller the bin size, the slower is the convergence of the algorithm. After considering all this points, we have chosen the bin size of the density to be  $\phi_\rho = 10^{-3}$ . The bin size of the order parameter, introduced in Chapter 5, has to be determined accordingly.

In the simulations, the histogram is generally checked each  $N_{\text{Sweeps}}$  number of steps whether the algorithm has already tunneled through the phase transition. The choice of  $N_{\text{Sweeps}}$  depends on the size of the simulated system. For small systems like  $N = 3^2 = 9$  we used  $N_{\text{Sweeps}} = 10^4$ , as the tunneling times were about  $\sim 10^5$  MC updates. For large systems like  $N = 25^2 = 625$ , we used  $N_{\text{Sweeps}} = 10^7$ , as in this case the tunneling time was much larger, like  $\sim 10^9$ . After  $N_{\text{Sweeps}}$  number of steps we check whether the highest state (top bin of the investigated density parameter range) has already been visited i.e. is not equal to one.

### 3.2.6 Mixing the system

In order to start the system in a general state in terms of the positions of the disks, we started by setting the radius to  $r_D = 10^{-3}$ , to start from the lowest density and then move randomly chosen disks for  $N_{\text{initial}} = 10^4$  times. This has been done by the application *Initializer*, which saved the state of the system afterwards, such that the main simulation *DiskSimulation* could be run.

### 3.2.7 Overlapping check

To check, if any two disks overlap, one could cycle through all the disks and examine if the current disk overlaps with any other after each displacement. Instead of this simple method we have used a grid cell method which divides the system into rectangular cells of the size a bit larger than the maximal disk

diameter (see Fig. 5.1). In a system of width  $L$  and  $N$  disks we used the following grid cell sizes:

$$C_{\text{width}} = \frac{L}{N-1}, C_{\text{height}} = \frac{L\sqrt{3}}{2(N-1)}.$$

Now we have to check overlapping of the moved disk with disks in its direct neighborhood only, given by the eight surrounding cells and the cell itself<sup>9</sup>. By setting up a list of neighbors for each disk at the beginning of the simulation, we can update the respective lists after every disk displacement. The computational expenses are significantly reduced in large systems by using this method. A nicer method would be to use the Voronoi [13] construction, which defines a non rectangular cell for every disk and makes it necessary to check the neighbor cells (the disks) only.

### 3.2.8 Scripts and additional code

#### Job submitting

To submit jobs to the hreidar cluster, a shell script has been written submitting the jobs and checking regularly whether a new job has to be sent to hreidar. Problems arise with the hreidar cluster as some jobs were killed before they were intended to. By restricting the simulation time to 2000 seconds, most of the jobs finished correctly. One problem with the shell scripts running on the login server is, that the more scripts we start, the more CPU is used on the login server itself, which should be avoided. We tried to submit a shell script itself to hreidar (what actually worked), but couldn't fix the problem with reading and writing data between the simulation runs. So we stuck with the former method.

## 3.3 Results

### 3.3.1 Density of states

As a first application of the proposed Wang-Landau algorithm we calculated the density of states for a system of  $N = 9$  disks<sup>10</sup>. We checked the histogram after every  $N_{\text{Sweeps}} = 10^4$  whether flatness has been achieved. The maximal displacement of a disk and the maximal radius change have been chosen the same as for the tunneling times ( $\Delta = 0.2$ ,  $\delta = 0.05$ ) and the flatness criterion has been set to  $H_{\text{min}}(\rho) > 0.8\bar{H}(\rho)$ , which means that if the minimum value in the histogram array is not less than 80% of the average, flatness is fulfilled. In Fig. 3.2 one can see the density of states for the investigated system with an example of the histogram as inset.

<sup>9</sup>Any other disk in any other cell can't overlap with the respective disk as we have chosen the grid cells to be a bit larger than the maximal disk diameter.

<sup>10</sup>A system consisting of nine disks is not really meaningful, but larger systems would have spent too much time to finish the simulation.

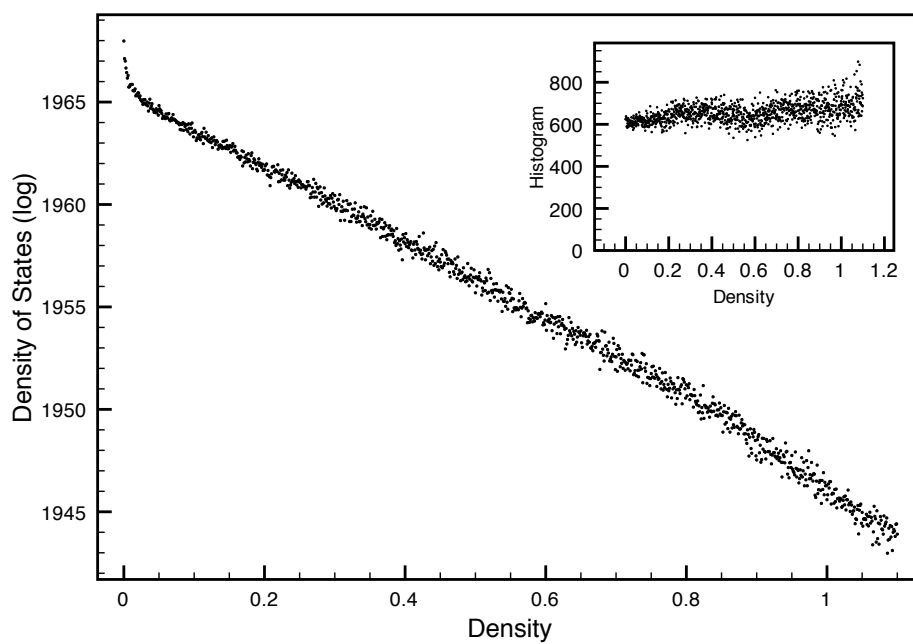


Figure 3.2: The (logarithm of the) density of states in a system of nine disks in dependence of the investigated parameter, the density. The highest values can be found at lowest densities while the lowest values are located around the maximal density (closest packing of the disks). In the inset one can see an example of a histogram which satisfies the flatness condition. For this simulation a flatness condition of  $x = 0.8$  has been used. To obtain the true density of states one would have to exponentiate and normalize the data.



## Chapter 4

# Tunneling times

The proposed Wang-Landau algorithm is one attempt to overcome the critical slowing down in the vicinity of a first-order phase transition. What we are interested in is what time it takes the algorithm to tunnel through the region of the phase transition where for normal MC simulations the probability of visiting such a state is exponential small ( $\sim \exp(-N)$ ). In this chapter we finally show that the tunneling times scale linearly in the size of the system  $\tau \sim N$ . For this purpose, we measured the tunneling times (actually we counted the number of sweeps) for systems of size  $N = 9, 16, 25, 36, 49, 64, 81, 100, 144$  several times<sup>1</sup>. Much larger systems should be simulated to obtain more exact results. We took different  $N_{\text{Sweeps}}$  for different system sizes as otherwise for large systems with small  $N_{\text{Sweeps}}$  the algorithm would have taken too long to finish. The results we obtained can be seen in Fig. 4.1. We used the discussed parameters for the simulation ( $\Delta = 0.2$ ,  $\delta = 0.05$ ) and let the algorithm tunnel from the liquid state to the solid state with density  $\rho = 1.1$ . This corresponds to a density in relation to the maximal density of  $\rho_{\text{rel}} \approx 95\%$ . From the obtained data we can conclude that the tunneling time is linear in the number of disks under investigation

$$\tau(N) = a \cdot N + b,$$

where  $\tau$  denotes the tunneling time,  $a$  the slope,  $b$  the shift parameter and  $N$  the size of the system in number of disks. Before we started to tunnel to states as high as the one discussed, we also measured the time the algorithm needed to reach lower states. In Fig. 4.2 one can see the tunneling times for different system sizes with the same parameters as before with the only exception, that the algorithm measured times from the liquid phase to the state with  $\rho = 0.9$ . The obtained parameters can be seen in Table 4.1. Actually one should simulate into the perfect crystal state ( $\rho = 2/\sqrt{3}$ ), where one expects the dependence to be exponential and not linear.

---

<sup>1</sup>We intended to simulate larger systems ( $N = 225, 400, 625$ ) but there arose some problems, which couldn't be solved anymore.

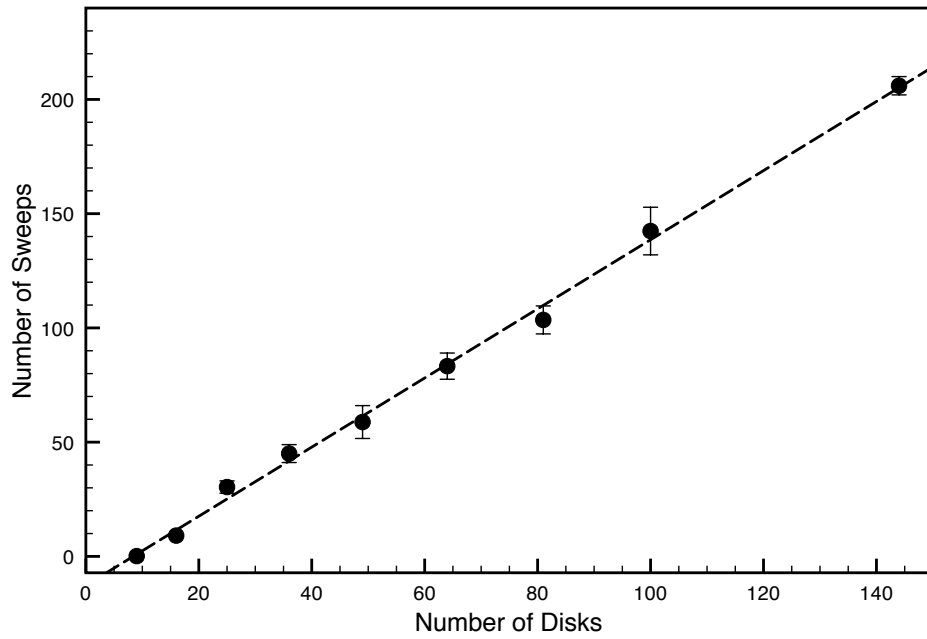


Figure 4.1: The tunneling times through the first order phase transition from the liquid to the solid phase for the investigated system. The linearity of the tunneling times  $\tau(N)$  in dependence of the system size  $N$  can clearly be seen. The number of sweeps are in millions ( $10^6$ ).

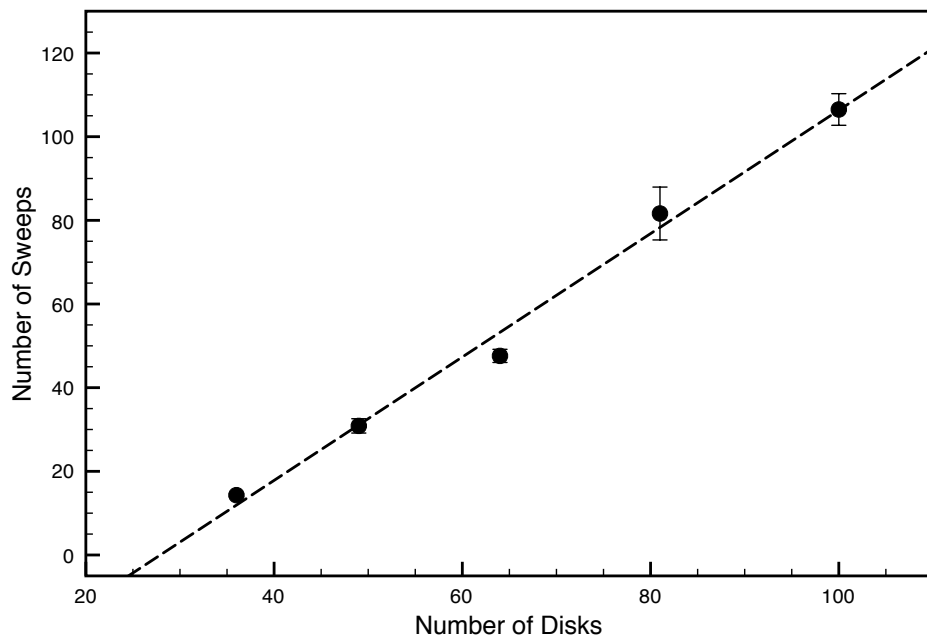


Figure 4.2: This tunneling times correspond to the case where the algorithm goes almost into the perfect crystal. The number of sweeps are in millions ( $10^6$ ).

Parameter	$\rho_{\max} = 1.1$	$\rho_{\max} = 0.9$
$a$	$1.51294 \cdot 10^6$	$1.47496 \cdot 10^6$
$b$	$-12.6924 \cdot 10^6$	$-41.1671 \cdot 10^6$
rms	3.45381	3.11631
$\chi^2$	1.03038	0.630162

Table 4.1: The parameters found for the tunneling times  $\tau(N) = a \cdot N + b$ , where  $a$  is the slope and  $b$  the shift in  $y$ -direction. The root mean square deviation (rms) and the chi-square ( $\chi^2$ ) are also listed here. We note that the slope in the first case, where we go into the crystal, is steeper than the one where we only go through the phase transition.



## Chapter 5

# Order parameter

The discussed Wang-Landau algorithm has been used to determine the density of states for a variety of systems. There arise some problems though while applying the algorithm to the systems of hard disks. In our system the particle moves are purely driven by entropy, as there is no energy dependence in the density of states  $\Omega$ . In order to speed up the algorithm one can think of helping the system to prefer certain moves. To achieve this, one can introduce an order parameter  $\phi$ , which causes the particles to move with higher probability to states of higher order. We chose an orientational order parameter  $\phi$  of the system, which is defined as

$$\phi = \left| \frac{1}{N} \sum_k \frac{1}{n_k} \sum_j \exp(6i\theta_{kj}) \right|^2,$$

where  $N$  is the number of disks in the system,  $n_k$  is the number of nearest neighbors<sup>1</sup> of disk  $k$ ,  $\theta_{kj}$  is the angle between disk  $k$  and  $j$  with respect to a fix axis, which we have chosen to be the  $x$ -axis. The first sum is over all disks  $k$  and the second sum is over all neighbors of disk  $k$ , called  $j$ . On the right side of Fig. 5.1 one can see in the upper left a local state that has a high order parameter and in the lower right a state with a low order parameter. The orientational order parameter lies in the interval  $\phi \in [0, 1]$ , where  $\phi = 1$  corresponds to the state of closest packing. The probability for an update from  $(\rho_1, \phi_1)$  to  $(\rho_2, \phi_2)$  for the Wang-Landau algorithm is given by

$$p((\rho_1, \phi_1) \rightarrow (\rho_2, \phi_2)) = \min\left(\frac{\Omega(\rho_1, \phi_1)}{\Omega(\rho_2, \phi_2)}, 1\right).$$

where the density of states  $\Omega$  depends now not only on the density  $\rho$  but on the orientational order parameter  $\phi$  as well. This means that we now have not only a one dimensional density of states but a two dimensional one.

The implementation of this entropy driven order parameter should speed up the algorithm, especially the tunneling times. We played around with the order parameter but can't give any results determining the influence of the additional parameter in comparison to the normal case with only the density as parameter.

---

<sup>1</sup>This is not the same as the list of neighbors we used in the case of the density as parameter alone. For convenience and simplicity we only want next neighbors (see 5.1).

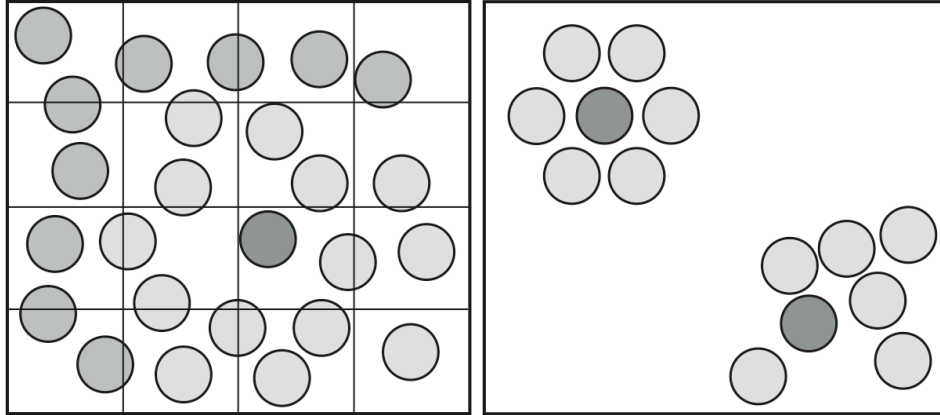


Figure 5.1: On the left one can see how the grid cell method works for a system of  $N = 25$  disks. The cell's width and height have to be larger than the maximal disk diameter. Taking the darkest disk as reference point, we have to check the disks in neighboring grid cells and the cell itself only. This is especially efficient in very large systems. Only disks whose center is in such a cell are considered because there is no way any other disk overlaps, as we have chosen the cell's dimension accordingly. For this special case, we don't have to take into account the periodic boundary conditions, but for all the boundary grid cells we have to. A visualization of a high order parameter and a low order parameter can be seen on the right. While the state in the upper left has a high local order parameter, the state in the lower right has a low order parameter. Local because to obtain the whole order parameter we have to sum over all disks in the system (global). The angles are measured with respect to the horizontal axis and with respect to the periodic boundary conditions.

## 5.1 Implementation

The implementation of the order parameter has to be done carefully. First of all we have to find a suitable bin size, which was chosen to be  $\phi_{\text{bin}} = 0.2$ . The flatness condition has to be rewritten in order to take the order parameter into account. In the one dimensional Wang-Landau algorithm we had the following criterion

$$H_{\min}(\rho) > x\bar{H}(\rho).$$

The natural generalization for the two dimensional case reads

$$H_{\min}(\rho, \phi) > x\bar{H}(\rho, \phi),$$

where  $\bar{H}(\rho, \phi)$  is the average over the whole 'matrix'  $H(\rho, \phi)$  and  $H_{\min}(\rho, \phi)$  is the minimum of  $H(\rho, \phi)$ . One problem that arises using this flatness condition is the following: The condition is only applicable if all the states can be reached. Otherwise the non reachable states lead to a never satisfied flatness condition<sup>2</sup>. We can neglect the non reachable states but this is not quite a decent solution. Another possibility is to use the standard deviation criterion

$$\sigma_H < x\bar{H}(\rho, \phi),$$

<sup>2</sup>This problem arises only, if one simulates the whole parameter space  $(\rho, \phi)$ . If we are only interested in a part of it, the criterion is quite useful.

where  $\sigma_H$  is the standard deviation, which has to be smaller than  $x\bar{H}(\rho, \phi)$ , a fraction of the mean of  $H$ . The fraction is around  $x \in [0.1, 0.3]$ , depending on the problem at hand. Another possibility is to satisfy a modified flatness criterion for the columns or rows only. The condition is then given by

$$H_{\min}^i(\rho, \phi) > x\bar{H}^i(\rho, \phi),$$

where  $i$  denotes a row or a column.

In order to calculate the orientational order parameter we have to find the nearest neighbors. This should not be confused with the list of neighbors we used in the case of the density parameter. Instead of taking only the nearest neighbors  $n_k$  of the disk  $k$  into account for the calculation of the order parameter, one could take all disks in the whole system and still would obtain the same result. While this global calculation is really slow for large systems, the local order parameter calculation is much faster and should be taken for simulation. Instead of taking all the disks in neighboring grid cells<sup>3</sup>, we only take disks  $j$  in one of the neighboring cells into account, whose distance to the investigated disk  $k$  satisfy

$$|r_k - r_j| < 2 \cdot r_{\text{initial}} + \epsilon = 1 + \epsilon,$$

where  $\epsilon$  is chosen to be 0.2 and  $r_{\text{initial}} = 1/2$ .

## 5.2 Results

The implementation of the (orientational) order parameter was verified by simulating the density of states for fixed density. According to Jooyoung and Strandburg [14], we knew what the density of states has to look like. We simulated a system of  $N = 16$  disks with maximal disk displacement of  $\Delta = 0.2$  and flatness condition of  $x = 0.8$ . The order parameter was in the range  $\phi \in (0, 0.94]$  and the bin size was chosen to be  $\phi_{\text{bin}} = 0.2$ . We refined the initial modification factor  $m = \exp(2)$  by  $m \leftarrow \sqrt{m}$  until  $m < \exp(10^{-4})$ . As this simulation ran at fixed density, we fixed the radius of the disks at the beginning of the calculation. We made simulations at densities of  $\rho = 0.7, 0.8, 0.85$  and  $0.9$ . Calculating the corresponding radii by

$$r_D = \frac{\sqrt{\sqrt{3}/2\rho}}{2}.$$

This yields the respective values  $r_D = 0.389, 0.416, 0.429$  and  $0.441$ . The density of states for a system consisting of 16 disks can be seen in Fig. 5.2. In some cases - depending on the fixed density - one can observe a maximum.

---

<sup>3</sup>If we talk of all neighboring grid cells, we always include the one cell the disk itself is in. We have to take not only the eight surrounding cells into account but nine of them.

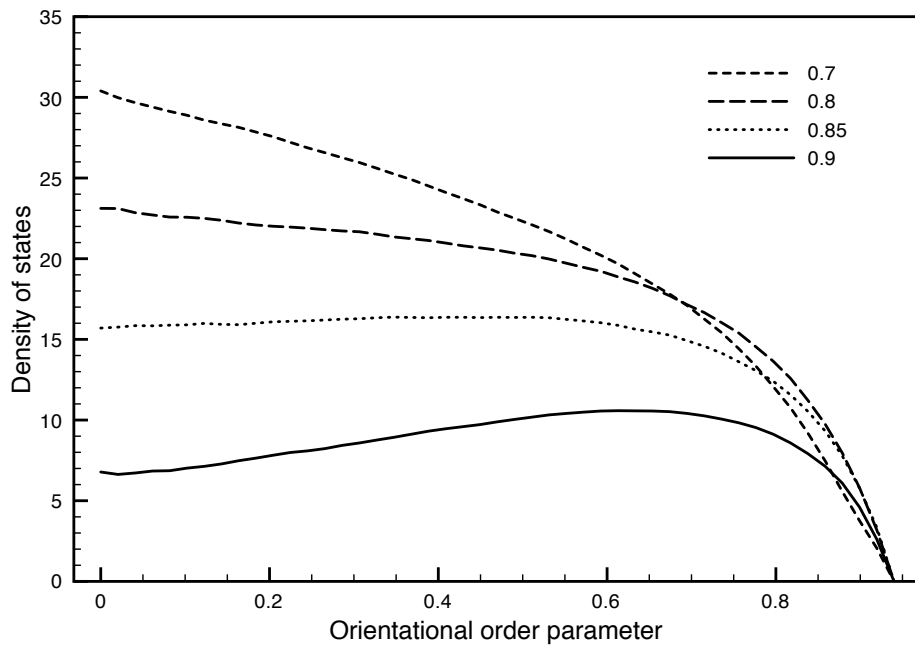


Figure 5.2: The density of states for different fixed densities in order parameter direction in a system of 16 disks. The  $y$ -axis is not the true density of states nor the one we obtained directly from the simulation. The  $y$ -axis is rescaled differently for the different curves to fit them all into the same window and be able to compare them. One can clearly see that there emerges a maximum around  $\rho = 0.85$  as further investigations would show.

## Chapter 6

# Conclusion and Outlook

The solid-liquid phase transition in a system of hard disks has been under investigation for several years. The slowing down of ordinary MC simulations in the vicinity of a first-order phase transition has been overcome by implementations of the Wang-Landau algorithm, which visits states with equal probability and lets the investigator obtain the density of states directly. In this report, we investigated the tunneling times  $\tau$  through the phase transition and determined the dependence of this tunneling times on the system size. The results show clearly that the dependence is linear. To get even better results, it would be necessary to simulate larger systems.

Additionally, we started with implementing a better update procedure by introducing an order parameter, which allowed the disks to prefer certain moves, i.e. moves to higher order. The next step would be to investigate the tunneling times considering the entropy driven order parameter updates as well and determine, whether the tunneling times can be shortened by this modification.

As we didn't simulate the phase transition into the perfect crystal, future investigations should go up to highest densities. There, one expects the transition time to be exponential in the size of the system. The benefits of the order parameter would be much greater in such simulations than it would be in the one we proposed.

Optimizing the code is possible in several ways; First, one could try to implement a local order parameter update. After every MC step we calculated the order parameter globally, thus needing the most CPU for this specific task. As we only move one disk at the time, the change in the order parameter is also locally and only changes for the moved disks and its neighbors. This implementation speeds up the code a lot. The simulations presented in this report only used one node at the time on the cluster computer. By using a parallel implementation of the code, like MPI, the simulations would not only be finished in a shorter time, but also use less IOstreams on the login server. Furthermore the used shell script could be optimized, as the current one is surely not the niftiest.

## Acknowledgments

First of all I would like to thank Prof. M. Troyer for giving me the possibility of making my diploma thesis in computational physics, helpful inputs and answers. With thankfulness I am also looking back to the comments and answers I got from G. M. Sigut, which helped me to get along with the hreidar cluster. To Andreas Ahlm I owe thanks for many shared hours during this thesis and playing the guitar in the sun. Finally, I am very grateful to my family, whose support I could always count on and whose motivation helped my finishing my studies.

## Appendix A

# Visualize disk system

In order to have some additional control over the overlapping and moving disk method, we implemented a simple method that draws the box and the disks into a window. The drawing routine does not take into account the boundary conditions. It is necessary to open x11 first and run the code after. We use the simple CImg library [12].

Initialization is done through

```
cimg_library::CImg<unsigned char> visu(430,373,1,3,1);
cimg_library::CImgDisplay draw_disp(visu,"Disk System & Frame");

F.DrawFrame(F.DiskSystem, F.getRadius(), visu, draw_disp);
```

Here F is the investigated system consisting of the box and the disks. The *DrawFrame* routine looks as follows

```
void Frame::DrawFrame(std::vector<Disk> &D, double Radius,
                      cimg_library::CImg<unsigned char> &visu,
                      cimg_library::CImgDisplay &draw_disp)
{
    const unsigned char red[3]={255,0,0};
    visu.fill(0);

    for (unsigned int i = 0; i < D.size(); ++i)
    {
        visu.draw_circle(int(floor(D[i].Position[0]*100)),
                        int(floor(D[i].Position[1]*100)),
                        int(floor(100*Radius)), red);
    }

    visu.display(draw_disp);

    while (draw_disp.closed == false)
    { ; }
}
```

The main drawing is made in the *for loop*, where step-by-step every disk's position and radius is sent to the CImg drawing routine. The while loop at the end ensures that the code does not continue without the drawing window being closed first.

# Bibliography

- [1] K. Strandburg, *Rev. Mod. Phys.* **60**, 161 (1988).
- [2] J. M. Kosterlitz and D. J. Thouless, *J. Phys. C* **6**, 1181 (1973).
- [3] S. T. Chui, *Phys. Rev. Lett.* **48**, 933 (1982).
- [4] M. A. Glaser and N. A. Clark, *Adv. Chem. Phys.* **83**, 543 (1993).
- [5] F. G. Wang and D. P. Landau, *Phys. Rev. E* **64**, 056101 (2001).
- [6] F. G. Wang and D. P. Landau, *Phys. Rev. Lett.* **86**, 2050 (2001).
- [7] R. H. Swendsen and J.-S. Wang, *Phys. Rev. Lett.* **58**, 86 (1987).
- [8] U. Wolff, *Phys. Rev. Lett.* **62**, 361 (1989).
- [9] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. M. Teller, and E. Teller, *J. Chem. Phys.* **21**, 1087 (1953).
- [10] <http://www.asgard.ethz.ch/>.
- [11] <http://www.boost.org/>.
- [12] <http://cimg.sourceforge.net/>.
- [13] D. Fraser and M. Zuckermann, *Phys. Rev. A* **42**, 3186 (1990).
- [14] J. Lee and K. J. Strandburg, *Phys. Rev. B* **46**, 17 (1992).