Max Planck Graduate Center
mit der Johannes Gutenberg-Universität Mainz

Benjamin J. Block

Nucleation Studies on Graphics Processing Units

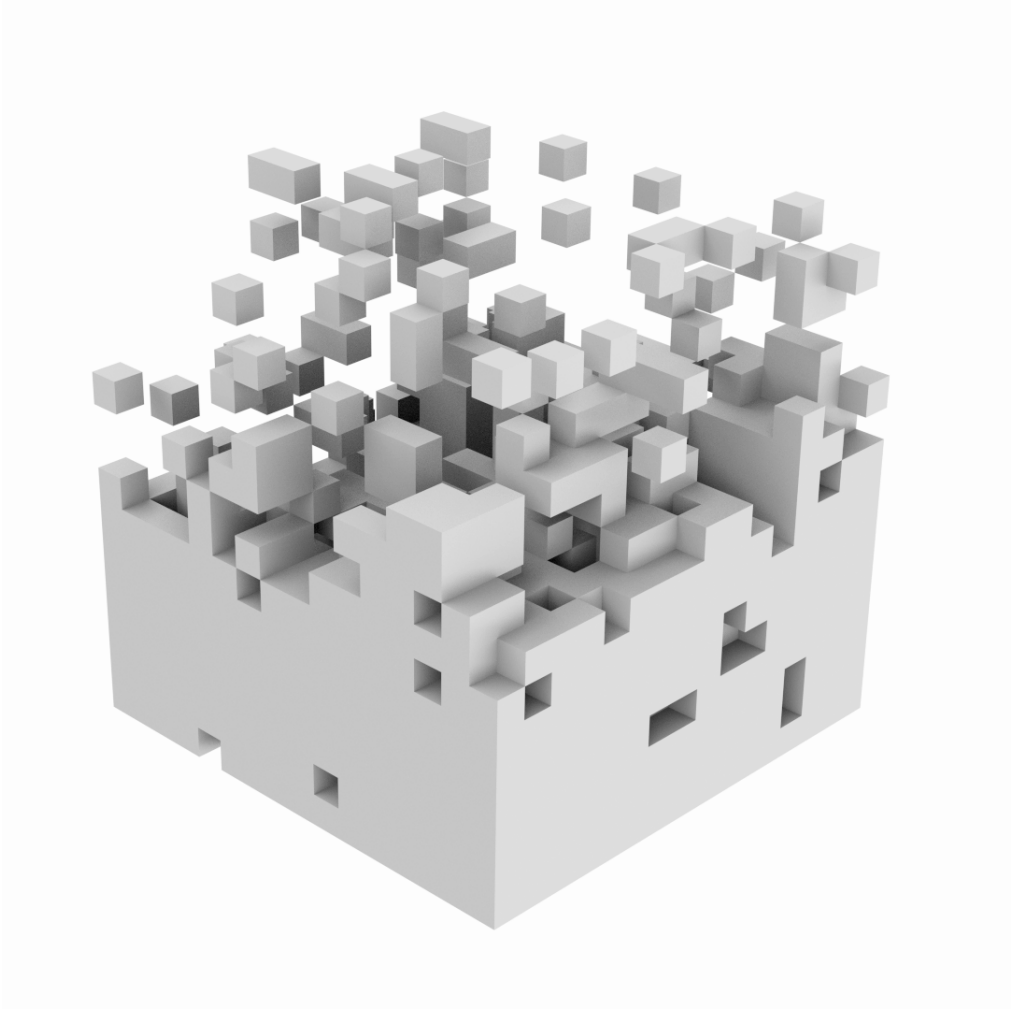
– Dissertation –

zur Erlangung des Grades eines `Doktor rerum naturalium (Dr. rer. nat.)'
der Fachbereiche:

- 08 - Physik, Mathematik und Informatik
- 09 - Chemie, Pharmazie und Geowissenschaften,
- 10 – Biologie,
- Universitätsmedizin

der Johannes Gutenberg-Universität.

Datum der Prüfung: 03.11.2014



Nucleation Studies on Graphics Processing Units

— Dissertation —

Benjamin J. Block
geb. in Idar-Oberstein

Mainz, im April 2014

Abstract

A system in a metastable state needs to overcome a certain free energy barrier to form a droplet of the stable phase. Standard treatments assume spherical droplets, but this is not appropriate in the presence of an anisotropy, such as for crystals. The anisotropy of the system has a strong effect on their surface free energies at low temperatures, while this effect is less important above the roughening transition temperature T_R. A simple model that features such an anisotropy is the Ising model. We perform large scale simulations of the Ising model to overcome finite-size effects and statistical inaccuracies. The scale of the simulations that are needed to produce meaningful results led us to the development of a versatile and scalable simulation code which can be used across different parallel computation devices such as graphics processing units (GPUs). Platform independence is achieved through abstract interfaces that hide platform specific implementation details. We prepare a system geometry that allows for the investigation of a flat interface with a tunable angle to the crystal plane, which touches an external wall. The contact angle Θ can be adjusted via a surface field H. A differential equation describing the behavior of the surface free energy in the presence of anisotropy for our system is discussed. Combined with thermodynamic integration methods, this equation is used to integrate the anisotropic surface tension over a large range of temperatures from well below T_R up to the vicinity of the bulk critical temperature T_C and is compared with prior predictions. Comparison with previous measurements in different geometries and with different methods shows good agreement and accuracy, which is achieved especially through the ability to simulate much larger systems than was possible in previous studies. The temperature dependence of the surface stiffness κ above T_R is extracted by measuring the curvature of the surface free energy near $\Theta = 90^\circ$. This measurement is comparable to the simulation data obtained in the literature and is in fact in better agreement with theoretical predictions regarding the scaling behavior of κ. We develop a low temperature model to explain the small angle behavior of the system far below T_R, where the angle stays virtually zero up to a critical field H_C, which only has a small system size dependency. When this critical field is overcome, the angle increases rapidly. H_C is linked to the *Step Free Energy*, which allows us to to analyze the critical behavior of this quantity. The effect of the hard wall has to be incorporated into the investigation. By comparing free energies at different system sizes, we are able to extract the free energy contribution of the contact line as a function of Θ. The temperature dependence is investigated by repeating this analysis at different temperatures. In the last chapter, a parametric simulation of 2D flow phenomena is accelerated using GPUs which can be used to simulate dynamics e.g. in the atmosphere. In particular we implement a parallel Evolution Galerkin operator and obtain a significant speedup in comparison to a serial implementation.

Zusammenfassung

Ein System in einem metastabilen Zustand muss eine bestimmte Barriere in der freien Energie überwinden um einen Tropfen der stabilen Phase zu formen. Herkömmliche Untersuchungen nehmen hierbei kugelförmige Tropfen an. In anisotropen Systemen (wie z.B. Kristallen) ist diese Annahme aber nicht angebracht. Bei tiefen Temperaturen wirkt sich die Anisotropie des Systems stark auf die freie Energie ihrer Oberfläche aus. Diese Wirkung wird oberhalb der Aufrauungstemperatur T_R schwächer. Das Ising-Modell ist ein einfaches Modell, welches eine solche Anisotropie aufweist. Wir führen großangelegte Simulationen durch, um die Effekte, die mit einer endlichen Simulationsbox einhergehen, sowie statistische Ungenauigkeiten möglichst klein zu halten. Das Ausmaß der Simulationen die benötigt werden um sinnvolle Ergebnisse zu produzieren, erfordert die Entwicklung eines skalierbaren Simulationsprogramms für das Ising-Modell, welcher auf verschiedenen parallelen Architekturen (z.B. Grafikkarten) verwendet werden kann. Plattformunabhängigkeit wird durch abstrakte Schnittstellen erreicht, welche plattformspezifische Implementierungsdetails verstecken. Wir benutzen eine Systemgeometrie die es erlaubt eine Oberfläche mit einem variablen Winkel zur Kristallebene zu untersuchen. Die Oberfläche ist in Kontakt mit einer harten Wand, wobei der Kontaktwinkel Θ durch ein Oberflächenfeld eingestellt werden kann. Wir leiten eine Differenzialgleichung ab, welche das Verhalten der freien Energie der Oberfläche in einem anisotropen System beschreibt. Kombiniert mit thermodynamischer Integration kann die Gleichung benutzt werden, um die anisotrope Oberflächenspannung über einen großen Winkelbereich zu integrieren. Vergleiche mit früheren Messungen in anderen Geometrien und anderen Methoden zeigen hohe Übereinstimung und Genauigkeit, welche vor allem durch die im Vergleich zu früheren Messungen wesentlich größeren Simulationsdomänen erreicht wird. Die Temperaturabhängigkeit der Oberflächensteifheit κ wird oberhalb von T_R durch die Krümmung der freien Energie der Oberfläche für kleine Winkel gemessen. Diese Messung lässt sich mit Simulationsergebnissen in der Literatur vergleichen und hat bessere Übereinstimmung mit theoretischen Voraussagen über das Skalenverhalten von κ. Darüber hinaus entwickeln wir ein Tieftemperatur-Modell für das Verhalten um $\Theta = 90^\circ$ weit unterhalb von T_R. Der Winkel bleibt bis zu einem kritischen Feld H_C quasi null; oberhalb des kritischen Feldes steigt der Winkel rapide an. H_C wird mit der freien Energie einer Stufe in Verbindung gebracht, was es ermöglicht, das kritische Verhalten dieser Größe zu analysieren. Die harte Wand muss in die Analyse einbezogen werden. Durch den Vergleich freier Energien bei geschickt gewählten Systemgrößen ist es möglich, den Beitrag der Kontaktlinie zur freien Energie in Abhängigkeit von Θ zu messen. Diese Analyse wird bei verschiedenen Temperaturen durchgeführt. Im letzten Kapitel wird eine 2D Fluiddynamik Simulation für Grafikkarten parallelisiert, welche u. a. benutzt werden kann um die Dynamik der Atmosphäre zu simulieren. Wir implementieren einen parallelen Evolution Galerkin Operator und erreichen eine signifikante Beschleunigung gegenüber einer seriellen Implementierung.

Contents

Introduction 9

1 The Ising Model and Nucleation Phenomena 13
1.1 Ising Model . . . 13
1.2 Homogeneous Nucleation . . . 15
1.3 Surface Anisotropy . . . 16
1.4 Heterogeneous Nucleation . . . 27
1.4.1 Wetting . . . 27
1.4.2 Line Tension . . . 30
1.5 Nucleation in the Atmosphere . . . 32

2 Methodology 37
2.1 Importance Sampling . . . 37
2.2 Boundary Conditions . . . 38
2.2.1 Hard Walls and Wall Fields . . . 38
2.2.2 Periodic Boundary Conditions (PBC) . . . 40
2.2.3 Antiperiodic Boundary Conditions (APBC) . . . 40
2.2.4 Generalized Antiperiodic Boundary Conditions (GAPBC) . 40
2.3 Simulations on GPUs . . . 43

3 GPU Implementation of the Ising Model 47
3.1 Introduction . . . 47
3.2 GPGPU History and CUDA . . . 48
3.3 The Ising Model on the GPU . . . 50
3.3.1 Checkerboard Update . . . 51
3.3.2 Simple Scheme in 2D . . . 52
3.3.3 External Fields . . . 54
3.3.4 Extraction of Magnetization . . . 55
3.3.5 Cluster Updates . . . 55
3.3.6 Random Number Generation . . . 57
3.3.7 Optimizing Memory Lookup . . . 58
3.3.8 Revised Memory Layout and Update Scheme . . . 61

3.3.9 Multi-Spin Coding . . . 64
3.3.10 Multi-GPU Implementations . . . 64
3.4 Validation . . . 66
3.5 Performance . . . 66
3.6 Canonical Simulation . . . 68
3.6.1 Fisher-Yates Shuffle . . . 69
3.6.2 Ergodicity . . . 71
3.7 Hardware Abstraction . . . 71
3.7.1 Kernel Definition . . . 73

4 Interfaces in the 3D Ising Model 75
4.1 Introduction . . . 75
4.2 Interfacial Free Energies . . . 75
4.3 Free Energy Integration . . . 78
4.3.1 With Respect to Temperature . . . 78
4.3.2 With Respect to the Wall Field . . . 81
4.4 Contact Angle . . . 83
4.4.1 Derivation of Young's Equation . . . 83
4.4.2 Without Young's Equation . . . 88
4.5 Surface Tension Anisotropy . . . 91
4.6 Surface Stiffness . . . 98
4.6.1 A Different Approach . . . 101
4.7 Low Temperatures . . . 101
4.7.1 A Model for Inclination Angles at Low Temperatures . . . 104
4.8 Scaling Behavior . . . 112
4.9 Line Tension Contribution . . . 116

5 A Finite Volume Evolution Galerkin Method on the GPU 123
5.1 Introduction . . . 123
5.2 Notations in Fluid Dynamics . . . 124
5.3 Partial Differential Equation . . . 124
5.4 Conservation Laws in Inviscid Fluids . . . 125
5.5 Weak and Strong Solutions . . . 126
5.6 Euler Equations . . . 126
5.7 Numerical Solutions of Hyperbolic Equations . . . 128
5.8 Discontinuous Galerkin Method . . . 129
5.9 Multidimensional EG Operator . . . 131
5.10 Mesh Adaptivity . . . 134
5.11 Numerical Experiments . . . 135
5.12 GPU Implementation . . . 138

Conclusion **143**

Bibliography **156**

Introduction

Despite being a rather old and long standing problem in physics, the nucleation process has always raised interesting questions. Many of them are still to be answered. Nucleation describes the phenomenon that a metastable state starts to decay by a thermally activated process. The decay starts when a nanoscopically small domain of the stable phase is formed by a rare spontaneous fluctuation. Due to the (unfavorable) cost in surface free energy of the domain a rather high free energy barrier needs to be overcome. This is why these fluctuations are rare. Clarifying how this barrier arises and identifying the physical quantities that must enter a calculation of this barrier is one of the central tasks of nucleation theory.

Different forms of homogeneous as well as heterogeneous nucleation are topics of interest especially in atmospheric sciences. They play an important role in aerosol generation in clouds and haze or in air pollution such as smog and smoke plumes and in cloud formations. Unfortunately, understanding of nucleation processes is still poor in certain areas up to today.

In order to tackle the problem of nucleation free energy barriers systematically,

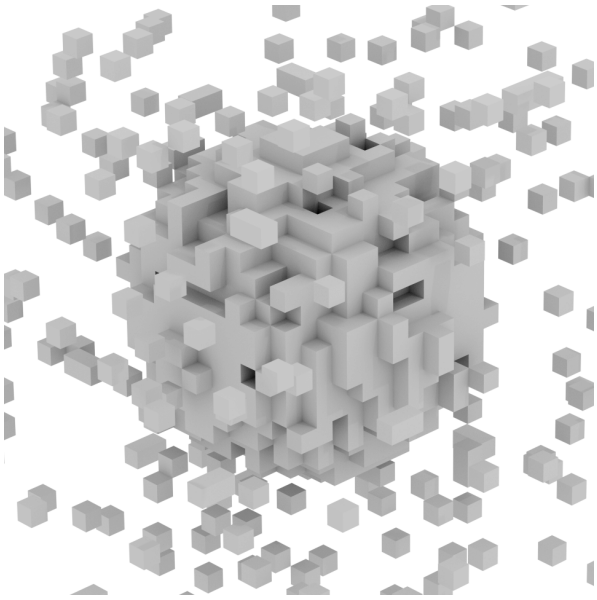


Figure 0.1: Simulating nucleation phenomena in the Ising model. Due to enough thermal fluctuations, a spherical droplet can form on an evenly spaced lattice.

it is useful to first start out with a very simple model and later improve on it step by step. Such a simple model is the lattice gas (or Ising) model, where particles may occupy the sites of a regular lattice. It is energetically favourable if two neighboring sites are occupied. Nucleation occurs at low enough temperatures, if the chemical potential μ of the particles exceeds the value μ_{coex} of vapor-liquid bulk coexistence. A droplet on the discrete and evenly spaced lattice of the Ising model is shown in figure 0.1.

Unfortunately, the Ising model did not get as much attention as it deserves in this area in recent years. A lot of theoretical understanding of the nucleation process and surface properties in general can be obtained by investigating this simple model. The Ising model belongs to the same universality class as the vapor-liquid transition in off-lattice fluids and as many binary systems in nature. Therefore, it has direct application to a range of natural phenomena.

In this thesis we do not look at the droplets that are formed in the nucleation process itself. We artificially stabilize a flat interface in a system in the two phase region to extract the properties of the interface that are important in the nucleation process. Under certain circumstances, such an interface — given the spacial discretization of the simulation model — undergoes a roughening transition at a roughening temperature T_R.

Rough interfaces are able to overcome the underlying lattice structure due to thermal fluctuations and can be aligned in any angle with the same free energy cost. Such an interface at an angle of nearly 45° to the lattice plane is shown in figure 0.2. Close to the critical temperature T_C, this rough interface has an isotropic surface tension $\gamma(\Theta, T)$. At low temperatures the interface has to follow the lattice planes to minimize free energy, so when approaching the roughening temperature T_R, the surface tension becomes increasingly anisotropic. Visible cusps are building up in the droplet shape.

Such a surface tension anisotropy is also present e.g. in ice crystals in the atmosphere. In the Ising model, the roughening temperature is at roughly half the bulk critical temperature. There is a large temperature window between them, which allows for a convenient investigation of the roughening transition independently from bulk critical phenomena in the Ising system.

The surface anisotropy of crystals can have a significant effect on the resulting nucleation rates which play a major role in the earth's atmosphere. Most of the participating molecules such as water form crystals in their solid state which are at temperatures way below their surface roughening temperature T_R. Most nucleation processes in nature are not homogeneous but involve nucleation agents. An example for such an agent is sulfuric acid in the atmosphere. These agents have a dramatic effect on the dynamics of the nucleation process itself. Also smoke or micrometer-sized dust particles can act as nucleation agents under

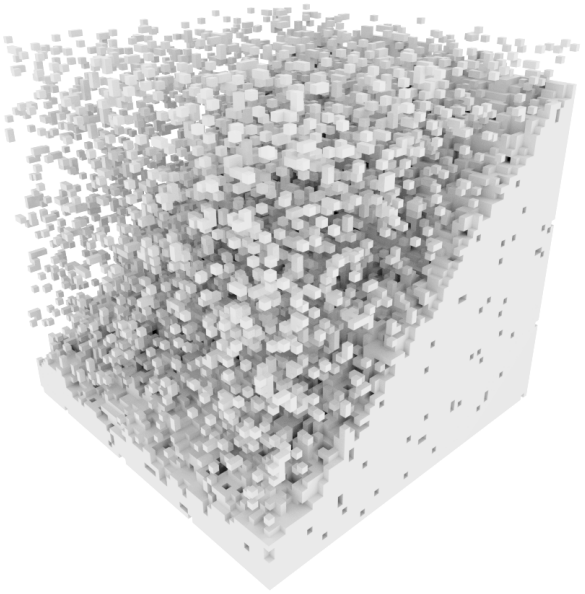


Figure 0.2: A *rough* interface between a dense and a dilute phase in a cubic simulation domain. The angle between the interface and a lattice plane can be adjusted using external fields.

certain conditions. Then nucleation involves the formation of (liquid or solid) droplets whose surfaces meet the substrate under an angle which is called the *contact angle*. As a first step to understand this process, we study a flat interface in contact with a hard wall at different contact angles.

The simulations that need to be performed require a large amount of computational resources. We have seen a steady decline in performance growth for serial microprocessor performance in recent years. To keep up with the growing demand for computational resources, there is a trend towards parallel computing architectures. We have developed a platform independent parallel implementation of the Ising model in two and three dimensions. Graphics Processing Units (GPUs) are cost effective parallel computation devices which can be used for general purpose computation tasks. This is why we used GPUs to produce simulation results at large system sizes that provide enough statistics for quantitative analysis of the roughening transition in the Ising model.

In chapter 1, the reader is introduced briefly to the theoretical background that is needed to tackle the questions answered in this thesis. The Ising model is defined, and its properties are discussed. Important concepts for the description of anisotropic interfaces and for heterogeneous nucleation processes are explained.

Chapter 2 introduces the methods used and gives a short introduction into the basics of Monte Carlo simulations. The means to stabilize an interface and create the configurations needed to study the interfacial properties are elaborated on. We use a custom kind of system boundary topology which is a generalized form of Antiperiodic Boundary Conditions (APBC) or more specifically: Antiperiodic

Boundary Conditions on a three dimensional Klein bottle. This topology reflects the symmetry of the system. We quickly realize that the computational effort to extract the needed properties is massive and cannot easily be tackled by conventional computer hardware.

This is the reason for chapter 3 where the means for acceleration of the computationally heavy parts are developed and presented which are needed to create an efficient GPU implementation of the 2D and the 3D Ising model.

In chapter 4, the implementation is used to simulate our model system and to extract new physical quantities, such as the contact angle in the model system as well as a contact angle dependent surface and line tension.

Chapter 5 is of a different nature. Here, a Finite Volume Evolution Galerkin scheme has been implemented on a GPU, for parametric simulation of 2D flow phenomena, which can be used to simulate dynamics e.g. in the atmosphere and in many other contexts.

Chapter 1

The Ising Model and Nucleation Phenomena

1.1 Ising Model

The Ising model is defined as an evenly spaced lattice of spins where each of the spins is in one of two possible states. In the simplest version, each spin can only interact with its nearest neighbors in each spacial dimension. The bulk interaction model can be defined through the Ising-Hamiltonian

$$H_{\text{Ising}} = -J \sum_{\{i,j\}} \sigma_i \sigma_j - H \sum_i \sigma_i \tag{1.1}$$

where J is the bulk coupling constant ($J > 0$ for a ferromagnet, $J < 0$ for an anti-ferromagnet) between two neighboring spins in the lattice, $\sigma_i \in \{-1, 1\}$ denotes the state a spin at lattice position i can be in, and $\{i, j\}$ means that the sum goes over all neighboring spins i, j. The second term describes a coupling to an external magnetic field H. We employ a simple cubic lattice, so in one spacial dimension there are two nearest neighbors, in two spacial dimensions there are four and in three spacial dimensions there are six nearest neighbors for each spin in the bulk.

The system has an analytical solution for a one dimensional lattice with external fields and a two dimensional lattice without external fields. However, the three-dimensional model (which will be used in this investigation) is only solvable numerically. For our simulations, we use the three dimensional Ising model. We will later incorporate external fields acting on the boundaries of the system and extend this Hamiltonian to account for a finite-size geometry. Specifically, we shall consider the case that external fields act only on particular surfaces of the system.

At high enough temperatures, there is no long range order in the system, independent of the system dimensionality. The 2D and 3D Ising-Model have a second-order phase transition at a critical temperature T_C, from a disordered to

one of two possible ordered phases. At the critical temperature, the spin correlation length diverges due to the formation of large clusters of two distinct phases (a majority up and a minority down spin phase). Above T_C, there is no distinction between those two phases. Without additional constraints, the system eventually demixes into one of the phases with equal probability, because there is a symmetry with respect to spin up/down exchange. For the 2D Ising-Model the critical temperature T_C can be calculated analytically as

$$T_{C,2D} = 2/\ln(1+\sqrt{2}) \approx 2.269$$

as derived by Onsager [1].

In the 3D Ising-Model, the first numerical value is given by Fisher [2] as early as 1967, which is an estimate obtained by high-temperature series expansions. The value obtained by the Monte Carlo study of Heuer [3] is

$$T_{C,3D,MC} \approx 4.5115(1).$$

Linked to the ferromagnetic phase transition at T_C are several critical exponents. These exponents are universal to many physical systems and characterize the singular behavior of physical quantities near the phase transition. This is called the universality hypothesis which states that these exponents do not depend on the microscopic details of the system. Newer, more accurate results on the critical temperature and critical exponents are found in [4, 5].

For this thesis, we focus on the three-dimensional case and since we need two coexisting phases to investigate an interface between them, we simulate wel below T_C, so long range fluctuations do only induce negligibly small finite-size effects.

In the following, we will sometimes call the two phases of the Ising model below the critical temperature L for liquid and V for vapor. This is in line with the well known interpretation of the Ising model as a *lattice gas*. The Ising Model can be used to study the interfacial behavior of gases and liquids, since the Ising model can be interpreted as a lattice gas of sites which can either be occupied or not. At each lattice site, an up spin ($\sigma_i = +1$) is interpreted as a hole, whereas a down spin ($\sigma_i = -1$) is interpreted as a gas molecule. The external field H is then proportional to the chemical potential of the particles. An interesting feature of this model is the symmetry under exchange of a gas particle and a hole on a lattice site. This is described in detail in [6].

In statistical mechanics, all discrete models (such as the Ising model) are necessarily anisotropic. This is handy, because in nature, many nucleating substances

are crystals and thus feature an anisotropy due to their lattice structure. As outlined in [6], the simplest microscopic model of crystal is a Kossel crystal [7]: Here, a crystal is a structure packed together out of rigid lattice cells. This proves to be a good model to obtain a qualitative picture, but in reality, it ignores many important effects of realistic crystals, such as lattice vibrations, an electronic structure and dislocations in the lattice. A typical two-phase equilibrium state has a dense phase which can at low enough temperatures be identified with the crystal phase and a dilute phase which consists of empty sites with a small concentration of gas particles. In the dilute phase, gas particles exist mainly as monomers or as very small clusters. In the microscopic picture, the crystal surface is the contour which separates the two phases. For sufficiently low temperatures, vacancies in the bulk of both the crystal as well as the dilute phase do not play a significant role. Before we further elaborate on the implications of anisotropic and the microscopic details of the surface, we introduce some basic concepts of the nucleation process under the assumption of an isotropic system.

1.2 Homogeneous Nucleation

Nucleation of a droplet of a phase L in a background of a supersaturated phase V is suppressed by a barrier in free energy ΔF that has to be overcome for a droplet to form (figure 1.1). If this barrier can be overcome by thermal fluctuations, spontaneous nucleation events can happen with a probability of

$$J \sim e^{\frac{-\Delta F}{k_B T}} \tag{1.2}$$

Assuming a spherical droplet of radius R and a clear-cut interface with zero thickness, this barrier ΔF is constituted of a volume and a surface contribution. The total excess free energy of the droplet according to *Classical Nucleation Theory* is

$$F_{\text{ex}} = \gamma \cdot 4\pi R^2 - \Delta p \cdot \frac{4}{3}\pi R^3, \tag{1.3}$$

where the first term is the surface contribution due to the surface tension γ and the second term is the volume contribution due to the pressure difference Δp inside the droplet. The volume free energy of the growing nucleus decreases the overall free energy, so a nucleation barrier results from a balance between the surface- and the volume free energy.

The surface tension always has to be positive, because otherwise the system would try to maximize the phase surface, effectively dissolving the phase L occupying the volume V in the surrounding phase V.

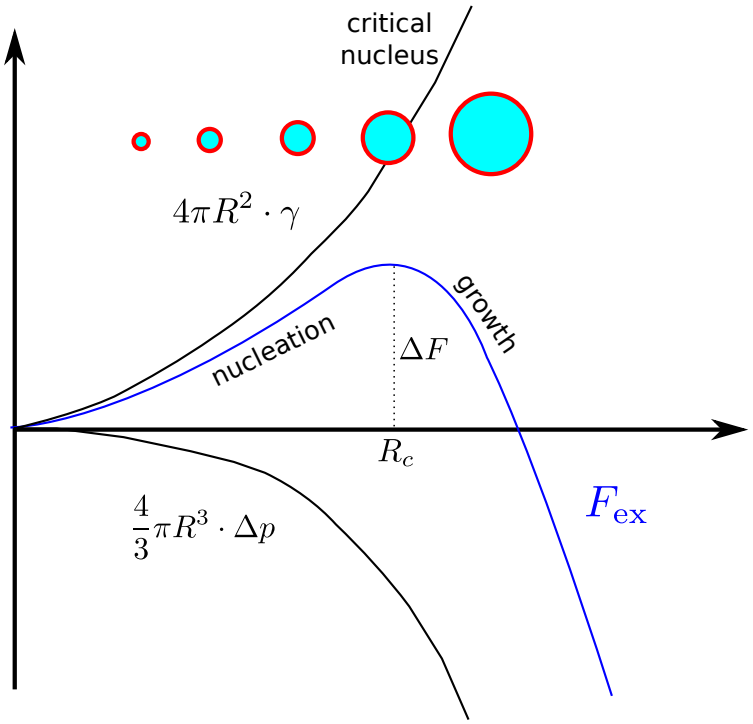


Figure 1.1: A spherical droplet according to classical nucleation theory has a surface and a volume contribution to the free energy, resulting in a nucleation barrier ΔF which needs to be overcome for a droplet to form.

Since surfaces are subject to fluctuations in terms of capillary waves as well as (dependent on the choice of ensemble) fluctuations of the position of the interface, they have an entropy which makes the free energy contribution due to the interface ($F_s = U_s - TS_s$) dependent on the temperature of the system. Here we disregard any problems in microscopically locating such an interface in a spin configuration of the Ising model [8].

1.3 Surface Anisotropy

So far (equation 1.3, figure 1.1) we have assumed a spherical droplet and a constant surface tension over the whole surface of the droplet. If the surface tension is not constant, we need to go a step further. Assuming that one bulk phase L occupies a region $V \in R^d$ in a surrounding bulk phase V, the corresponding surface contribution $F_s(V)$ to the free energy F is equal to the integral

$$F_s(V) = \int_{\partial V} \gamma(\Theta, \Phi) \mathrm{d}A \tag{1.4}$$

over the boundary ∂V of V, with the angles Θ and Φ being a parametrization of the orientation of the surface. Treating this surface integral analytically requires a profound knowledge of the nature of the coexisting bulk phases.

The nucleated droplet will take on the shape that will minimize the total free energy of the surface at a given volume. If this phase L is completely surrounded by phase V and both phases are fluid, they have an isotropic surface tension and the phase L will form a droplet of spherical shape.

However, if one of the phases is crystalline, the surface tension between the two phases can become anisotropic. The condition for this to happen is explained later in this section. This leads to the creation of flat faces and facets in the droplet shape. The general form of the droplet shape in a system with surface tension anisotropy can be derived with the *Wulff-Construction* [9]. This construction is a polar plot of the surface tension versus the orientation of the interface. A more recent overview of equilibrium crystal shapes can be found in [10]. For discrete simulation models as well as crystals, the surface tension γ is not expected to be isotropic and may vary with its angle to the lattice plane. In our investigations in chapter 4, we create a situation where there is only one tunable angle to a lattice plane, which coincides with the contact angle Θ of a flat interface to the wall surface.

For nucleation of ice crystals in the atmosphere, the problem of dealing with an anisotropic surface free energy (equation 1.4) clearly arises. However an atomistic description of the interaction between water molecules is a formidable problem and it is doubtful where one could make progress with the theory of ice nucleation without having dealt with simpler models earlier. This is one of the motivations to consider anisotropy effects for nucleation in the Ising model.

Because the simple Ising system offers an anisotropic surface tension $\gamma(\Theta)$, it is a very welcome tool to analyze the nature of the surface tension in anisotropic systems and its implications in general.

If overhangs in the surface shape can be neglected, the interface can be treated analytically in the *Solid On Solid* (SOS) model, which defines the surface as a height $h(x, y)$ over a two dimensional plane (x, y). For the Ising model, the SOS description can be derived by assuming the height z to be an integer value h_{ij} on a lattice (i, j). In many cases, the essential properties of the crystal surface are well described by this simplification, not only qualitatively but also quantitatively. With the simplification of using the Hamiltonian of the SOS model,

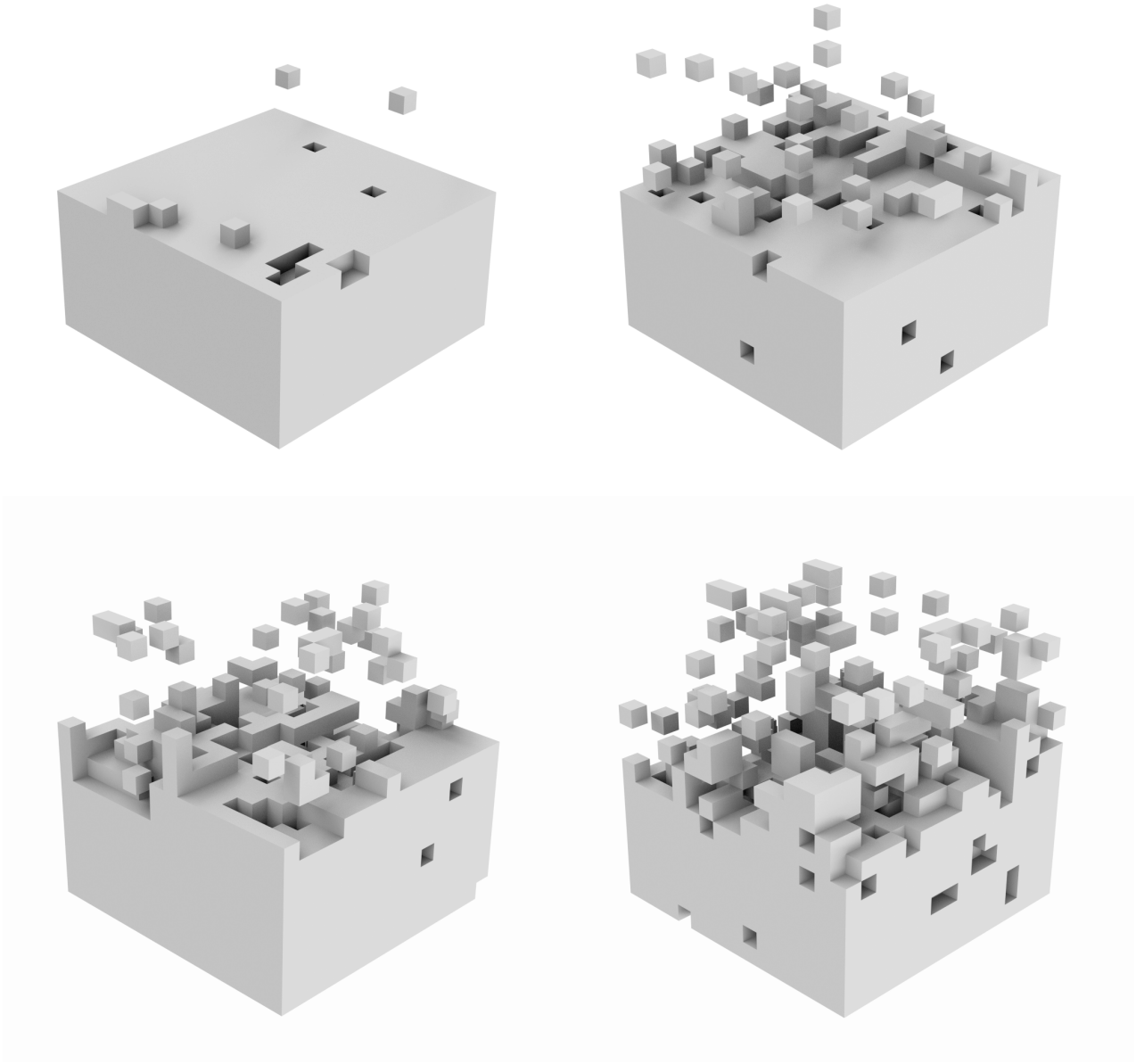


Figure 1.2: A $16 \times 16 \times 16$ surface slice of a system of size $120 \times 120 \times 120$ at different temperatures $T = 2.0$ (upper left), $T = 2.5$ (upper right), $T = 3.0$ (lower left) and $T = 3.5$ (lower right). Up spins are depicted as solid blocks while down spins are empty space, in analogy with a lattice gas model. The upper half of the cube therefore depicts a dilute phase (V), while the lower half depicts a dense phase (L). Because of the exchange symmetry there are as many holes in the lower half of the box (which cannot be seen in this visualization) as there are solid boxes in the upper part. The surface exhibits increasing roughness with higher T. Even at low temperatures there are already overhangs and islands in the surface, raising questions about the accuracy of the SOS approximation.

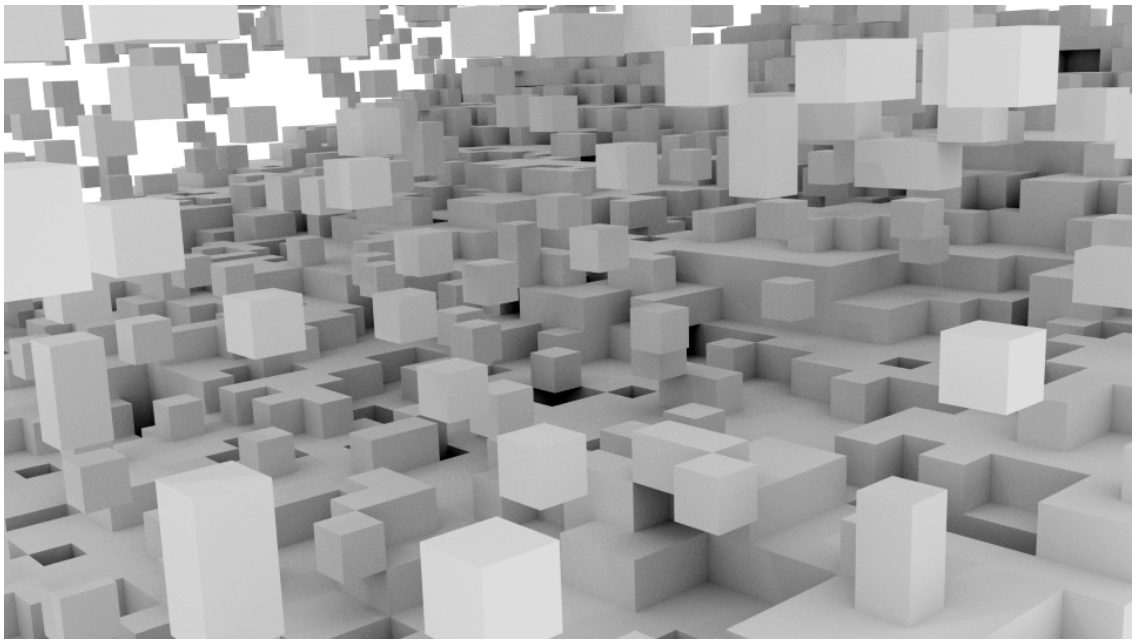


Figure 1.3: The surface of a tilted interface in the rough phase ($T = 2.6$). Due to enough thermal fluctuations, the shape of the surface is not governed by the underlying lattice structure. Below the roughening transition, the surface takes on a completely different structure (see figure 1.4). The picture is taken from a simulation of a system of size $184 \times 504 \times 504$.

the surface in the Ising model can be treated analytically as has been done in [11]. This simplification can break down if the existence of overhangs and islands above the surface have an impact on the results. The existence of these structures cannot be denied and they have to be expected especially at higher temperatures (figure 1.2).

Interfaces are called *rough* when they are able to overcome the lattice structure they are defined on due to entropy and can be aligned in any angle with respect to the lattice planes (figure 1.3). At low temperatures they are not able to overcome the lattice structure and they have to follow the lattice planes due to energy constraints. The interface which separates the two ordered domains (liquid and gas) can undergo a roughening phase transition at a finite temperature T_R from a rigid to a rough interface. Close to the critical temperature T_C, this rough interface has an isotropic surface tension $\gamma(\Theta, T)$, while when approaching the roughening temperature T_R, the surface tension becomes increasingly anisotropic, until visible cusps are building up in the droplet shape. In the two-dimensional Ising model, the roughening temperature T_R is at

$$T_{R,2D} = 0,$$

which means for finite temperatures it is always in the rough phase. Only at $T = 0$, cusps in the droplet shape can form.

In this context, we consider an interface with an average position which coincides with a lattice plane (e.g. normal to the z-axis). For the three-dimensional Ising model, the roughening temperature T_R was not well known [12], but estimated to be around $T_{R,3D} \approx 2.5$ for a long time. Later work [13, 14] was able to predict concrete values for T_R. In [14] the roughening temperature is located at

$$T_R \approx 2.45374.$$

Analysis of exactly solvable models [6] as well as renormalization group calculations [15] indicate that a roughening transition usually is an infinite order transition [16] with a very weak singularity in the free energy. Figure 1.4 shows a tilted surface well below T_R. The structure of the interface is fundamentally different from a system above T_R, featuring several kinks with plateaus inbetween. A term closely related to the roughening transition is the *Step Free Energy*, which can be used as an order parameter of the transition. Consider a flat interface between two phases. At $T = 0$ and $\Theta = 0$, the interface is perfectly flat. A flat interface with a single step over a distance L_x represents an interface with a small tilt of

$$\Theta = \arctan\left(\frac{1}{L_x}\right)$$

.

At low temperatures, a single kink looks like shown in figure 1.5. Such an interface can be treated in the SOS model, or with more accuracy in the full Ising model. Non-zero temperatures induce small thermal excitations at the interface just as in the bulk of the phases. While the bulk excitations grow to macroscopic size (which is controlled by the bulk correlation length) at the bulk critical temperature T_C, the interface thermal excitations exhibit critical behavior already at the temperature T_R. For $T < T_R$ the probability to create a step of length L orthogonal to a flat interface decreases proportional to

$$\exp\left(-\frac{L f_s(T)}{k_B T}\right) \tag{1.5}$$

where $f_s(T)$ is the free energy attributed to the step in the interface per unit length [17]. According to SOS models [16, 18] combined with a hyperscaling relation [6], the *Step Free Energy* takes on a nonexponential scaling behavior of

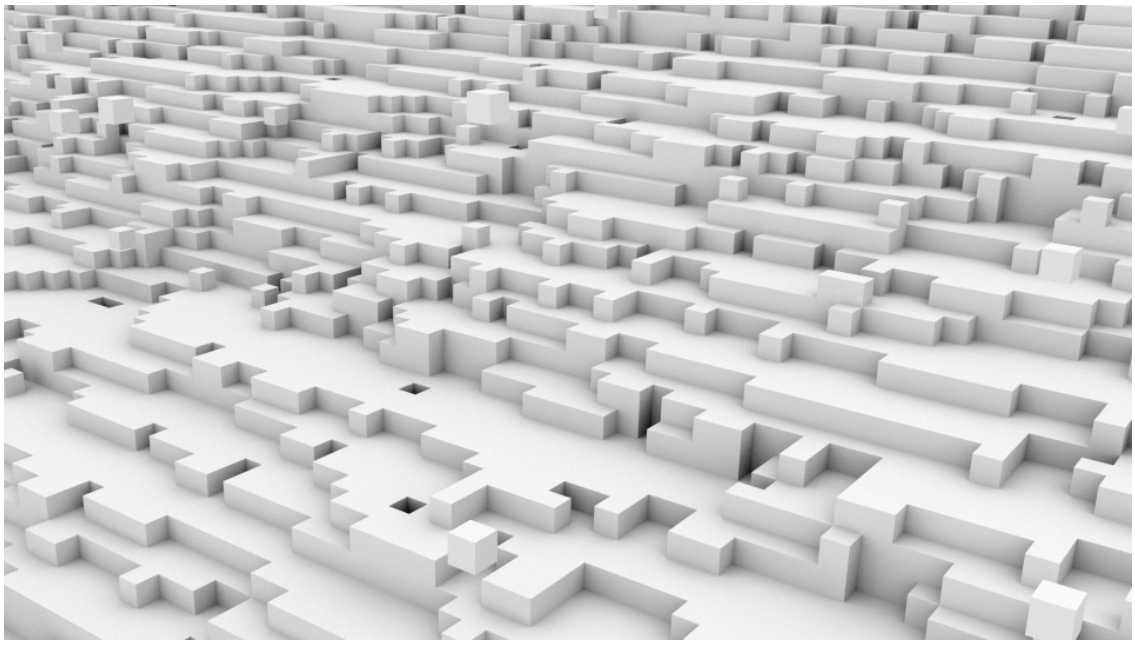


Figure 1.4: The surface of a tilted interface well below the roughening temperature ($T = 1.5$) has a fundamentally different structure of plateaus and kinks, where each kink can be attributed to a *Step Free Energy*. This example shows the surface of a system of size $184 \times 504 \times 504$ at a wall field of $H = 0.8$, which results in a contact angle of nearly $45°$.

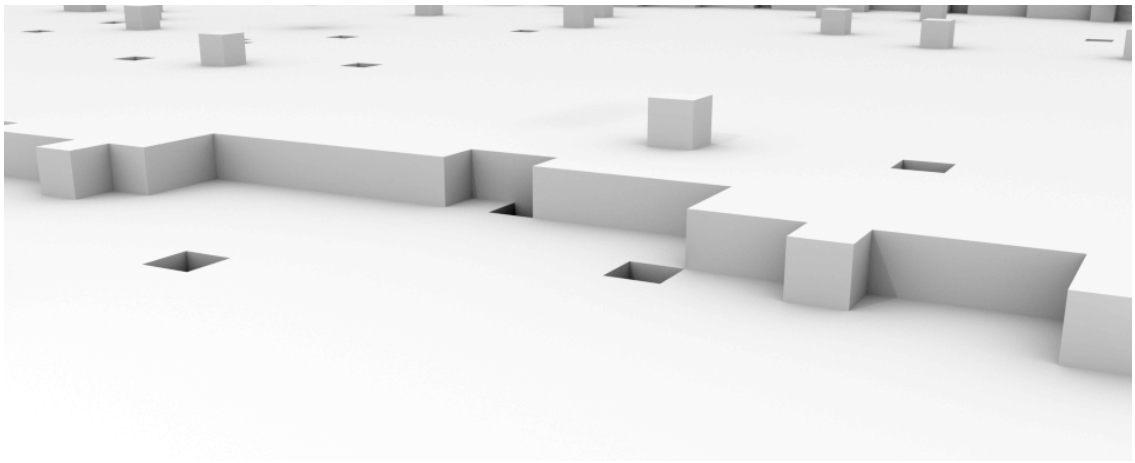


Figure 1.5: The same system as in figure 1.4, but with a smaller wall field $H = 0.645$. With decreasing wall field, the distance between successive kinks becomes larger, until there is only one kink left. For a single kink to form, the *Step Free Energy* needs to be overcome. At even lower fields, the kink-plateau structure disappears completely and the interface becomes completely flat.

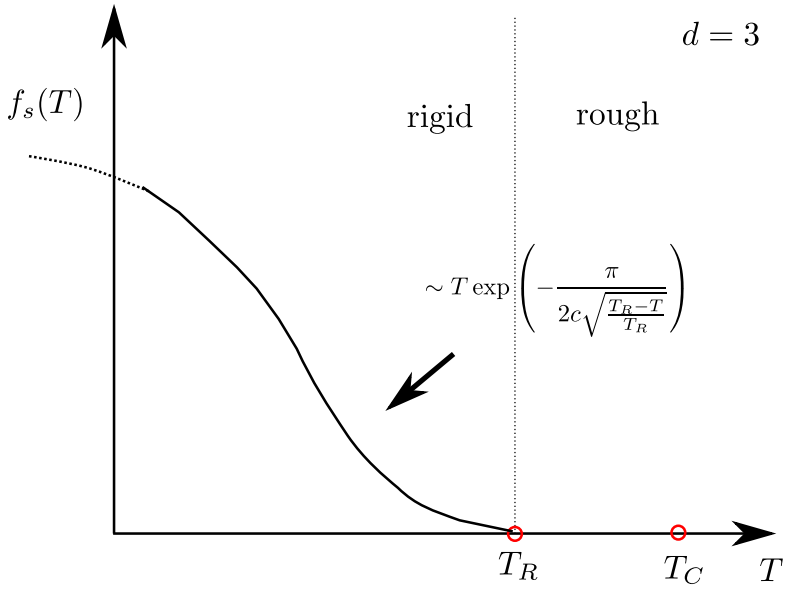


Figure 1.6: Below the roughening temperature T_R, a *Step Free Energy* appears, with a non-exponential temperature scaling behavior.

$$\frac{T}{f_s(T)} \sim \exp\left(\frac{\pi}{2c\sqrt{\frac{T_R - T}{T_R}}}\right) \quad , \quad T < T_R \tag{1.6}$$

which disappears at the roughening temperature T_R. After this *Step Free Energy* is overcome, the new and growing layer can be interpreted itself as a two phase lattice gas where one phase represents the growing layer of spins and the other phase represents the still unoccupied sites of the layer [19].

The transition itself can be characterized by the vanishing of the *Step Free Energy*, much like the bulk transition is characterized by the vanishing of the interfacial tension γ. Equation 1.5 indicates that at the interface, as soon as f_s vanishes, resistance against the formation of large clusters vanishes too, and a growth at all supersaturations of the unoccupied sites is happening. As it turns out, a new layer of spins can start to build up before the first layer is completed and close to the *Roughening Transition*, the growth usually occurs on more than one layer at the same time [6]. Fluctuations have an entropy which leads to a temperature dependence ($F = U - TS$). In a finite system, long range fluctuations get suppressed when the correlation length reaches the system size. The spectrum of fluctuations is cut off at the system size. This leads to a size

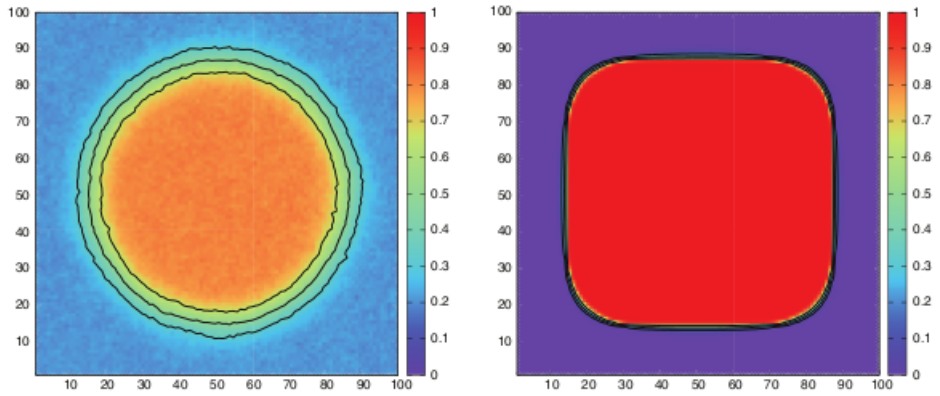


Figure 1.7: Droplet shapes in the 3D Ising model: Contours of constant density $\rho(x, y) = \rho_i$. At temperatures well above the roughening temperature ($T = 4.3$), droplets in the Ising model have a spherical shape (left). Well below the roughening temperature however ($T = 1.0$), they exhibit large anisotropy, thus forming a more and more boxlike shape (right). The images were taken from Schmitz et. al. [8].

Figure 1.8: Surface tension anisotropies lead to all kinds of droplet shapes in nature, such as this ice crystal. The crystal shape has a large impact on the dynamics of a nucleation process. The image was taken from the outstanding collection in [20].

dependence of the free energy in finite systems. This size dependency needs to be quantified by simulating different system sizes and analyzing the change in free energy between them. The form of a droplet in the Ising model at various temperatures has been studied by [8] (figure 1.7). Below T_R, the *Step Free Energy* leads to an anisotropy of the surface tension. This in turn leads to droplet shapes which are not spherical but can exhibit cusps and facets as observed in the shape of ice crystals (figure 1.8). Of course, for such crystal shapes also the anisotropy of the speed of crystal growth plays a major role. Unlike figure 1.7, figure 1.8 does not depict a crystal shape as it would result from the *Wulff construction*.

The interface has a certain resistance against small entropic variations of the surface angle Θ which is a fluctuating degree of freedom that leads to a contribution to the total free energy. This property of the interface is characterized by the *surface stiffness*. It is quantified by the stiffness coefficient κ.

If the surface is not flat and aligned to an axis plane, we need to write out the total interfacial energy as an integral of the surface tension over the whole surface area accounting for the changing surface tension at each surface point. In an a simple cubic lattice like in the Ising model, it makes sense to express the surface tension in dependence on the angles (Θ, Φ) to the lattice planes. For simplicity (and as given by our special geometry), we assume only a dependence on one angle Θ.

$$E_{\text{int}} = L_z \cdot \int \gamma(\Theta) \mathrm{d}l$$

We use a simple cubic lattice, which makes the definition of a tilt angle in relation to one lattice plane obvious. As explained in [21], if we assume the interface to be defined by a height profile $h(x)$ along the x axis, we can further write

$$E_{\text{int}} = L_z \cdot \int \mathrm{d}x \sqrt{1 + \left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)^2} \gamma(\Theta)$$

with

$$\sqrt{1 + \left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)^2} \approx 1 + \frac{1}{2}\left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)^2$$

and (note that $\Theta - \Theta_0 = \mathrm{d}h/\mathrm{d}x$)

$$\gamma(\Theta) \approx \gamma(\Theta_0) + \gamma'(\Theta_0)\left(\frac{\mathrm{d}h}{\mathrm{d}x}\right) + \frac{1}{2}\gamma''(\Theta_0)\left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)^2$$

The linear terms in $\left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)$ yield only boundary terms to the integral, so we arrive at

$$E_{\text{int}}/L_z = \gamma(\Theta_0) \int \mathrm{d}x + \frac{\gamma(\Theta_0) + \gamma''(\Theta_0)}{2} \cdot \int \mathrm{d}x \left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)^2 \tag{1.7}$$

This makes the definition of the stiffness coefficient κ as

$$\kappa = \gamma(\Theta_0) + \gamma''(\Theta_0)$$

obvious.

The importance of this coefficient becomes clear in the context of the capillary wave model (see [22], or [23] for an analysis in Nickel). There the dynamics of the surface is also formulated in the height variable $h(x)$ by defining the capillary wave Hamiltonian, in this case

$$H_{\mathrm{cw}} = \frac{1}{2} \int \mathrm{d}x \kappa \left(\frac{\mathrm{d}h}{\mathrm{d}x} \right)^2$$

See [24] for a renormalization group flow using block spin observables which can be directly measured with the Monte Carlo method. The surface stiffness has an interesting behavior when approaching the roughening temperature from above (see figure 1.9), where it gradually approaches a value of $\frac{\pi}{2}$ at T_R, and jumps to infinity directly below. Approaching T_R it features a scaling behavior [25] of

$$\beta\kappa(T) = \frac{\pi}{2} \left(1 - c \cdot \sqrt{\frac{T - T_R}{T_C}} \right) \quad , \quad T > T_R \tag{1.8}$$

where c is a non-universal constant. This behavior will be analyzed quantitatively in detail in section 4.6. It has been one of the aims of this thesis to show that the computational methods developed are suitable to characterize the properties of interfaces in anisotropic systems accurately, both below and above the roughening transition temperature.

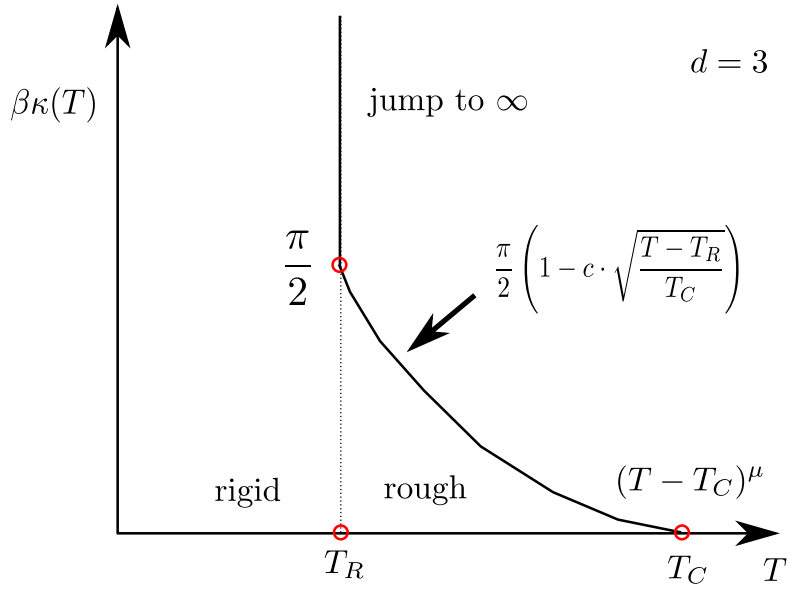


Figure 1.9: The surface stiffness coefficient of a 3D Ising lattice. When approaching the roughening temperature T_R from above, the surface stiffness rises up to $\frac{\pi}{2}$ at T_R, according to a non-exponential scaling law. Below T_R, it instantly jumps to infinity. When approaching the bulk critical temperature T_C, it has the same universal scaling behavior with the exponent $\mu = (d-1)\nu$ as the surface tension γ, since the γ'' contribution vanishes eventually.

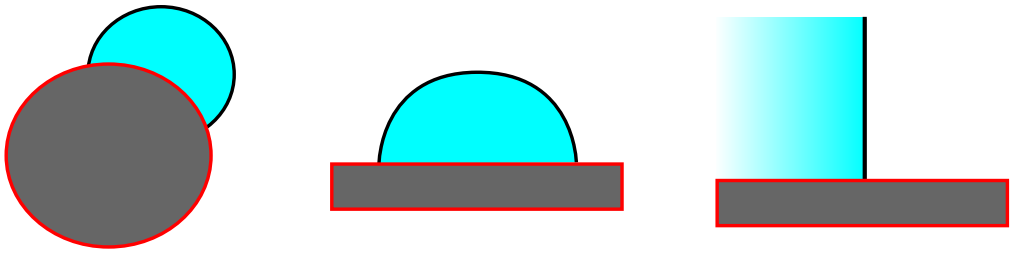


Figure 1.10: Heterogeneous nucleation can exist in different forms. A situation like the leftmost is the most general situation, with a curved droplet attached to a curved surface. This occurs in nature such as in nucleation on nucleation agents like SO_4 in clouds. The middle configuration has been investigated by [27] and the rightmost is a mostly synthetic situation that occurs in our system due to the choice of boundary conditions, with a flat interface in contact with a hard wall. The contact angle between the surface and the substrate is related to the surface tensions of the different phase boundaries by Young's equation.

1.4 Heterogeneous Nucleation

If heterogeneous nucleation is considered, the surface tension and thus the free energy that dictates the nucleation rate is influenced by a range of new phenomena that need to be considered separately. Figure 1.10 shows different kinds of nucleation phenomena occurring in nature of in artificially constructed systems. We consider again the droplet of the bulk phase L in a background V. Assuming an isotropic interfacial tension γ_{LV}, Young's equation [26]

$$\gamma_{LV} \cos\Theta = \gamma_{LW} - \gamma_{VW} \tag{1.9}$$

gives the contact angle as a function of the respective surface tensions between the phases and the planar wall γ_{LW} and γ_{VW} (see figure 1.11).

In contact with the wall, the droplet shape changes. To derive the final form of the droplet on a wall, the influence of the wall and the anisotropy of the surface tension have to be taken into account at the same time. A modified version of the *Wulff construction* to derive the equlilibrium shape of small droplets in contact with a foreign substrate is developed in [28]. For an illustration of the derivation see figure 1.12. For a treatment with gravity taken into account see [29].

1.4.1 Wetting

Young's equation (equation 1.9) defines the contact angle Θ of a droplet with respect to the wall to which it is attached.

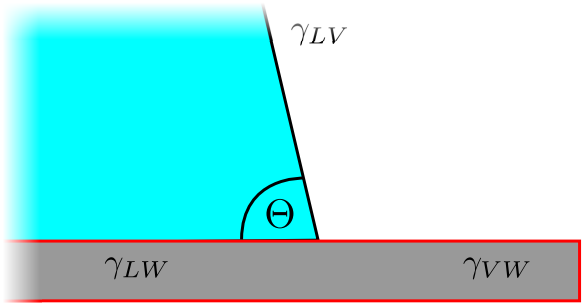


Figure 1.11: A two-phase interface with an angle Θ in contact with a wall. The angle Θ is determined via Young's equation via the surface tensions γ_{LV}, γ_{LW} and γ_{VW}.

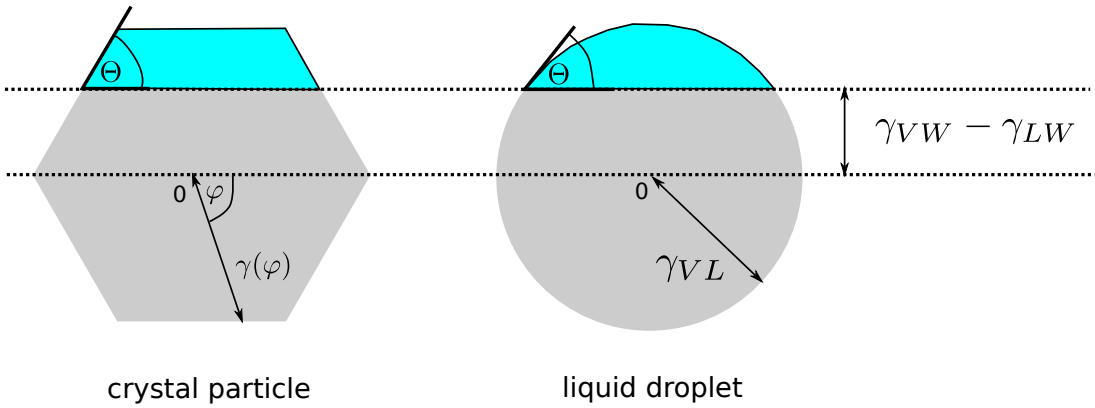


Figure 1.12: The *Wulff construction* is a powerful tool to construct the droplet shape even for a nucleus with surface anisotropy (left side). The *Wulff construction* is a polar plot of the surface tension $\gamma_{VL}(\varphi)$ in dependence of the orientation angle φ with respect to a lattice plane. A horizontal line is drawn that is shifted by the difference in surface tensions between the wall and the liquid and the wall and the vapor above the line of symmetry. The part of the droplet that is above the horizontal line is the droplet shape in contact with the wall. This procedure also works with an isotropic surface tension like the liquid droplet on the right, restoring the prediction of Young's equation 1.9.

It may occur that the nucleus spreads out to cover the wall by a thin film of phase L: This phenomenon is called *wetting*. In nature, it is brought forward from the difference in interactions between both phases and the wall and can have a tremendous effect on nucleation rates.

Because the cosine function has an upper bound of 1 (at $\Theta = 0$) and a lower bound of -1 (at $\Theta = 180°$), Young's equation gives a criterion to distinguish between 3 different regimes (see figure 1.13). We illustrate the different regimes along the lines of the explanation in [28].

The first one is called *complete wetting* and occurs for $\gamma_{LV} \leq \gamma_{LW} - \gamma_{VW}$. In this case, eq. 1.9 can not be fulfilled, so the system tries to maximize the contact area

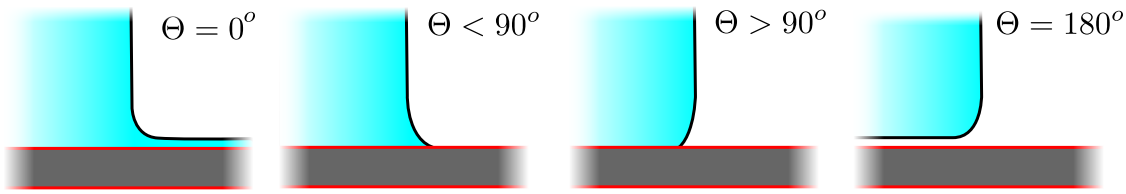


Figure 1.13: The case of a wall attached flat interface. From left to right, a *complete wetting* situation, two different partial wetting configurations and a configuration in the *complete drying* regime.

to the wall, leading to a complete film of L along the wall. This corresponds to a contact angle of $\Theta = 0$. In the *Wulff construction* (figure 1.12), this corresponds to the case that the crystal shape is completely below the horizontal line. If equation 1.9 can be fulfilled, which means

$$-\gamma_{LV} < \gamma_{LW} - \gamma_{VW} < \gamma_{LV}, \tag{1.10}$$

the system realizes the respective contact angle Θ in the isotropic case, and the crystal shape is predicted according to figure 1.12. The resulting shape above the horizontal line is the crystal shape that is realized by the droplet in the partial wetting regime. The contact angle Θ of the interface with the wall will be estimated as part of this work later, in section 4.4. An anisotropic generalization of Young's equation [11] will be used to derive the surface tension for different contact angles.

The last case is realized when $-\gamma_{LV} \geq \gamma_{LW} - \gamma_{VW}$ and is called *complete drying*. Then Young's equation also can not be fulfilled and the free energy is minimized when the droplet is not in contact with the wall at all. This corresponds to a contact angle of $\Theta = 180^\circ$. In the *Wulff construction* the crystal shape is completely above the horizontal line. In this regime it is the vapor (V) rather than the liquid (L) phase that is preferred by the wall i.e. the liquid avoids direct contact with the wall and between the wall and the liquid is an intruding vapor film.

The boundary between the wetting and the non-wetting regime is marked by a phase transition. If the boundary field is kept constant and one varies the temperature, one has *either* a wetting *or* a drying transition. The low temperature phase always exhibits *partial wetting*, while the high temperature phase exhibits either *complete wetting* or *complete drying*, depending on the direction of the surface field. [11] gives a description of the wetting transition in terms of the contact angle as the order parameter of the transition. This transition can be a first order transition [30, 31] ([32] for a treatment in Argon films). But as [30] revealed: In a mixture of two fluid phases close enough to their critical point, the contact angles will become zero against a surface of any third phase, which

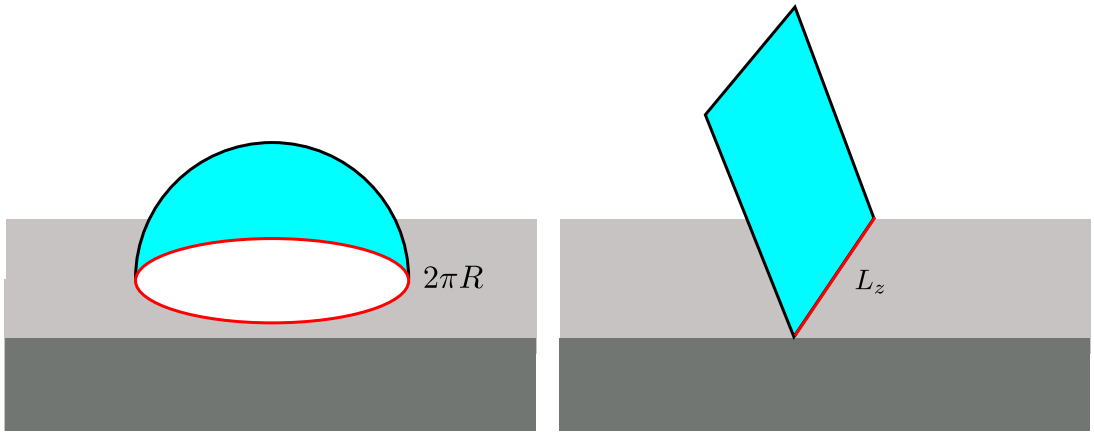


Figure 1.14: A two phase interface attached to a wall always forms a one dimensional three-phase boundary. The left figure shows a wall attached droplet, while the right figure shows a flat interface such as in our system.

is not involved in the critical point. One critical phase, always perfectly wets the extraneous phase. This phenomenon is called *critical wetting*. In our case, *critical wetting* occurs, so the transition from a wetting to a non-wetting state is of second order [33, 34, 35]. Recent work on the behavior of the critical wetting transition for short range forces has been done by Bryk et. al. [36]. When the surface field is varied, then one can have both a wetting and a drying transition in the same system, in dependence of the surface field.

1.4.2 Line Tension

For droplets in the nanometer regime an accurate thermodynamic description requires to account for not only the bulk and surface free energies but also for the special properties of the phases in the vicinity of three-phase contact. For a heterogeneous nucleation process on a wall, such as a wall-attached droplet, such a three phase boundary exists. This boundary is one dimensional in three dimensions and a point in two dimensional systems. In one dimension, a three phase contact cannot exist. It is assumed that this three phase boundary has an additional contribution to the free energy [37, 38, 39, 40]. In three dimensions, the line tension τ is this free energy per line length.

$$\tau = F_{\text{line}}/L. \tag{1.11}$$

This three phase interface has been investigated with Monte Carlo simulations in the Ising model with wall attached droplets in [27, 41] (figure 1.14 left) and for a flat surface by [42] (figure 1.14 right). This thesis will extend on the work of both [41] and [42]. It is not clear to which extent the line tension is dependent

on the curvature of the line, so in principle, the case of a liquid droplet in contact with a solid substrate and a planar liquid gas interface in contact with the substrate need to be treated separately. [37] indicates that curvature effects can be neglected when the droplet radius is larger than the thickness of the contact region, with the exception of small contact angles. In our case, a line tension contribution would be proportional to the linear dimension L_z, along which the interface is attached to the wall. If the surface is packed between two hard walls in one dimension, there are two lines that contribute to the free energy.

Of course, for the prediction of nucleation barriers due to wall-attached droplets the line tension in general has to be included, if droplets are in the nanometer size range.

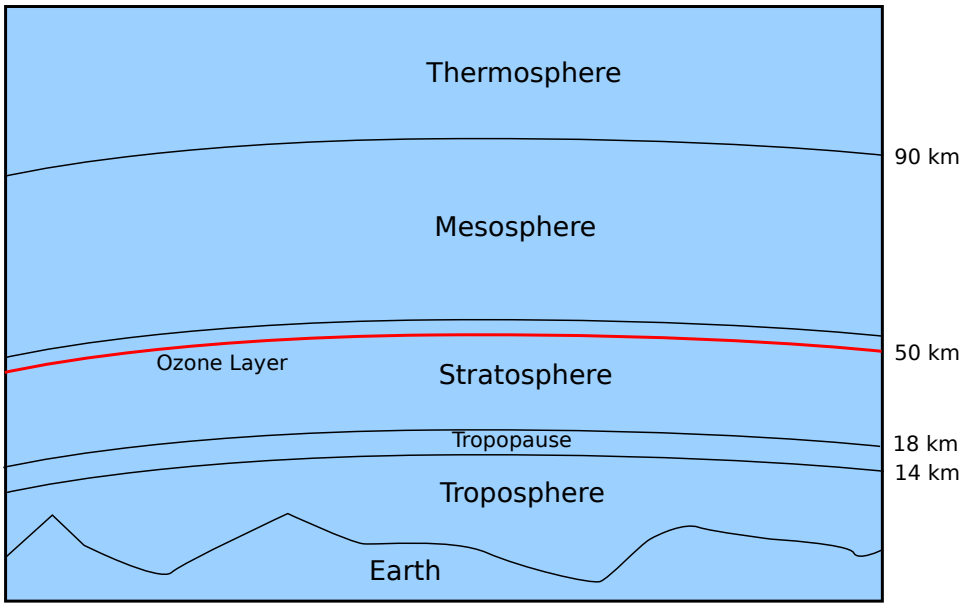


Figure 1.15: The layers of the earths atmosphere, from the boundary layer up to the thermosphere. Nucleation phenomena play an important role in several different places.

1.5 Nucleation in the Atmosphere

The nucleation process is important at different levels in the atmosphere, ranging from the planetary boundary layer (which is due to being the lowest part of the atmosphere, in the lower end of the troposphere, directly influenced by its contact with the planetary surface). The results of nucleation and growth of water droplets (or ice crystals) in the atmosphere are familiar to everybody (such as clouds, haze, et cetera). The nucleation phenomena are usually heterogeneous with nucleating agents such as aerosols, smoke particles, etc. An important distinction of nuclei within the atmosphere is between *aerosol particles*, which are mainly present in the troposphere and describe particles in the size range from a few to several hundred nanometers and *cloud particles* (hydrometeors) with can be up to several hundred micrometers in size. Aerosols have tremendous effects such as an influence on the earth's radiation budget directly by scattering solar radiation [43] or to induce cloud formation [44]. A large contribution to the particles in the atmosphere is due to nucleation from the surrounding gas phase [45, 46], but very few trace compounds out of the many different compounds that are found in the atmosphere are actually able to nucleate new particles. The key precursor compound is usually sulphuric acid (SO_4) from which binary nucleation is possible [47, 48, 49, 50], as well as ternary nucleation in combination with ammonia [51, 52] or organic acids [53] which can increase the nucleation

rate from sulphuric acid. It has been shown that, given the conditions in the middle and upper troposphere, the neutral nucleation of H_2SO_4 and H_2O is slower than ion-induced nucleation of H_2SO_4 and H_2O [50]. In coastal regions, other agents, namely oxidized ionide compounds are also important for nucleation. After nucleation the growth process is influenced by other organic acids with low vapor pressure. The nucleation processes in the atmosphere are very complex and numeric or even analytic treatment is not really possible. It is important to note that the chemical complexity of the atmosphere leads to the nucleation of droplets that are not only consisting of pure water, but of a solution of a variety of chemical components. The larger cloud particles have much different effects on the earths atmosphere and need to be treated differently from the smaller aerosol particles. The formation of cloud drops usually takes place in updrafts, which leads to a steady rise in relative humidity (RH) due to adiabatic expansion. At temperatures below $0°C$, cloud drops as well as the smaller aerosol particles can nucleate ice particles. The relative humidity is linked to the supersaturation (S_i) of the nucleating atmosphere via the equation

$$\frac{RH}{100} = S_i = a_w \exp\left(\frac{2\gamma_l}{k_B T n r}\right) \tag{1.12}$$

where a_w is the water activity of the droplet, r is the droplet radius and γ_i is the surface tension of the droplet and n is the number density of molecules in the liquid. Particle nucleation in a specific nucleation mode always occurs until the vapor uptake of the nucleated particles deplete the background supersaturation.

Generally, homogeneous nucleation of ice in the atmosphere can occur directly from the vapor. Vapor to ice nucleation however has a large energy barrier because of the large surface energy $\gamma_{\text{ice, vapor}}$. That is why ice nucleation usually takes place inside of liquid droplets which have been nucleated homogeneously beforehand, and the main attention lies on homogeneous freezing of liquid droplets during ascension in the atmosphere. A summary of different ice nucleation modes at different temperatures (e.g. heights in the atmosphere) is given in figure 1.16. Usually, water droplets do not freeze directly when cooled below $0°C$. Freezing has to be triggered by aerosol particles down to a temperature of about $-38°C$. At this temperature, freezing can occur without external influence [55]. A thorough review of laboratory experiments on ice nucleation on atmospheric aerosols can be found in [54]. Even though it has been indicated that the effect of variations in the density and surface tension models have been found not to cause significant changes in the nucleation rates [47], the effects of the anisotropy of the surface tension, which is expected from the

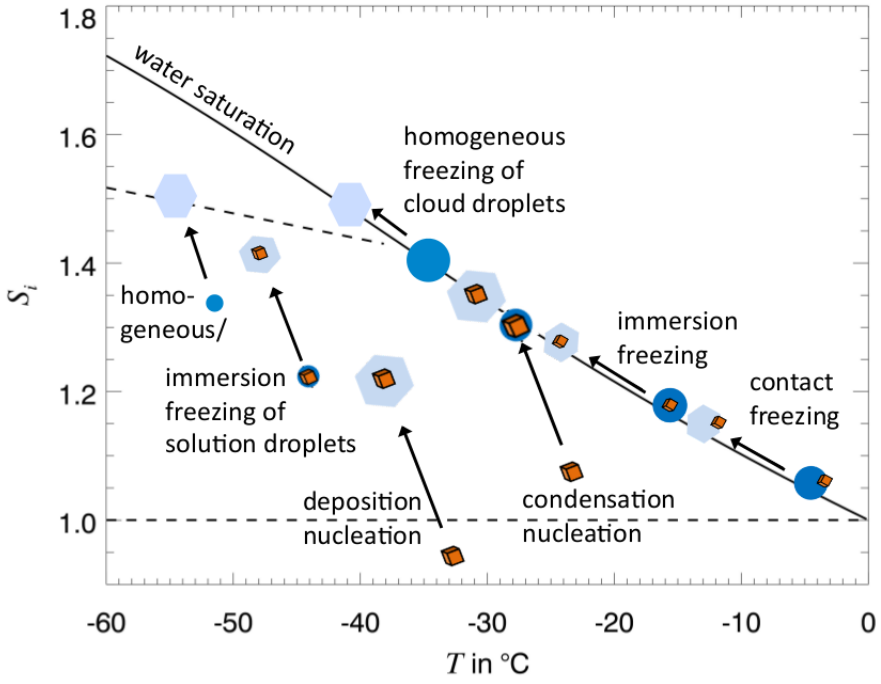


Figure 1.16: At different temperatures, different modes of nucleation are possible for ice crystals from water droplets and directly from the surrounding vapor, as well as heterogeneously from aerosol particles. The image was taken from [54].

crystalline structure of frozen water need to be investigated. Mason et. al. [56] find indication that the crystallographic structure of surfaces have an influence on ice nucleation probability. Simulations of the atmospheric processes are on vastly varying size and time scales as well as particle concentrations are usually relying on parametrizations and immense simplifications. Parametric simulations such as the Evolution Galerkin scheme that is discussed in the context of a GPU implementation in chapter 5 rely in turn on microscopic properties such as nucleation rates that can only be retrieved by microscopic treatments of the nucleation process itself.

Since we tackle the nucleation process from a microscopic point of view, other fundamental insights can be gained into the microscopic nature of the nucleation process, that can be supplied to a parametric simulation in turn. Classical nucleation theory is not applicable to the most of the nucleation processes, since the nucleus that is on the onset of forming a droplet does not form a body spherical body with a clear cut surface but is a result of spontaneous fluctuations of just a few molecules. An important aspect of the nucleation process that is discussed

in this thesis is the anisotropic nature of the surface tension in systems where translational and rotational invariance is broken. Water nuclei in the atmosphere form crystals, where special effects such an anisotropic crystal surface has to be taken into account when calculating properties such as the nucleation rate.

Since in experimental observations, the microscopic details of the nucleation process are not easily accessible, experiments mostly rely on the measurement of macroscopic quantities such as nucleation rates. In computer simulations however, we are able to observe a system at the microscopic level, extracting properties that can be used as input for theoretical models to predict experimentally observable macroscopic quantities and behavior. The methodology for these microscopic simulations is outlined in the following.

Chapter 2

Methodology

The dominating method for studies in the Ising model is the Monte Carlo method. A Markow process of first order is used to sample configuration space. Configurations are created successively by altering a configuration by an accepted Monte Carlo move. The Monte Carlo move we employ is a *flip*, where a single spin σ_i is replaced by $-\sigma_i$.

2.1 Importance Sampling

The Monte Carlo time t denotes the amount of Monte Carlo steps that have been taken along a Markov chain in configuration space. The transition probability of a configuration to go from a configuration x_i to another configuration x_j is denoted as

$$W_{ij} = a_{ij} w_{ij}, \tag{2.1}$$

where a_{ij} denotes the selection probability of a move and w_{ij} the acceptance probability. In the case of a checkerboard update, the selection is not stochastic anymore, but predefined. After an update of the white lattice sites follows an update of the black lattice sites with 100% probability.

To ensure convergence to a stable distribution, the total flux into state x_j needs to become zero for all i

$$\frac{\partial P(x_i, t)}{\partial t} = \sum_{j(j \neq i)} P(x_j, t) W_{ji} - \sum_{j(j \neq i)} P(x_i, t) W_{ij} = 0 \tag{2.2}$$

This is called *global balance* criterion. Since this criterion is hard to implement, one usually enforces an even stronger criterion called *detailed balance*, which is defined as follows:

$$P(x_i) W_{ij} \stackrel{!}{=} P(x_j) W_{ji} \tag{2.3}$$

The transition probability is chosen so that it fulfills *detailed balance.* One choice that fulfills this is due to Metropolis [57]

$$W_{ij} = \begin{cases} e^{-\beta \Delta E} & \text{for} \quad \Delta E > 0 \\ 1 & \text{else} \end{cases} \tag{2.4}$$

A move that leads to a lower configurational energy thus always gets accepted, a move that leads to a higher energy is accepted by a certain probability, which gets higher for higher temperatures. Since the evaluation of the Metropolis criterion is stochastic, for each evaluation, a random number has to be drawn (for technical implications see section 3.3.6).

2.2 Boundary Conditions

The finiteness of the system is addressed with specified boundary conditions. The behavior of the Ising System in two-phase coexistence investigated in this thesis is heavily dependent on the nature of the boundary conditions imposed on the system. In this section we discuss the various boundary conditions imposed on the simulation systems used in this thesis.

2.2.1 Hard Walls and Wall Fields

We consider a system of L_x lattice planes in x-direction. At the planes $x = 0$ and $x = L_x - 1$, we define a wall potential. The interaction potential at the boundary is replaced by a wall potential. We study the Ising model in a three dimensional simulation box with the exchange energy J in the bulk, and the energy $|H|$ due to the magnetic field in the surface planes. The total Hamiltonian will then be:

$$H_{\text{Ising}} = -J \sum_{\{i,j\}} \sigma_i \sigma_j - H \sum_{l=0,1} (-1)^l \sum_{\{k\}_l} \sigma_k, \tag{2.5}$$

where $\{k\}_l$ denotes all the spins in the layer next to the wall l, where l can be either 0 for the upper wall, or 1 for the lower wall. The field H is positive for the upper boundary layer and negative for the lower boundary layer. For simplicity, we are using $J = 1$. Since the Ising-Model has only short-ranged interaction and the field will only act on the first and last layer of spins, we are ignoring the effects of long-range nature van der Waals' forces which are known to differ in many respects from the short-range case of surface fields we are about to investigate [58].

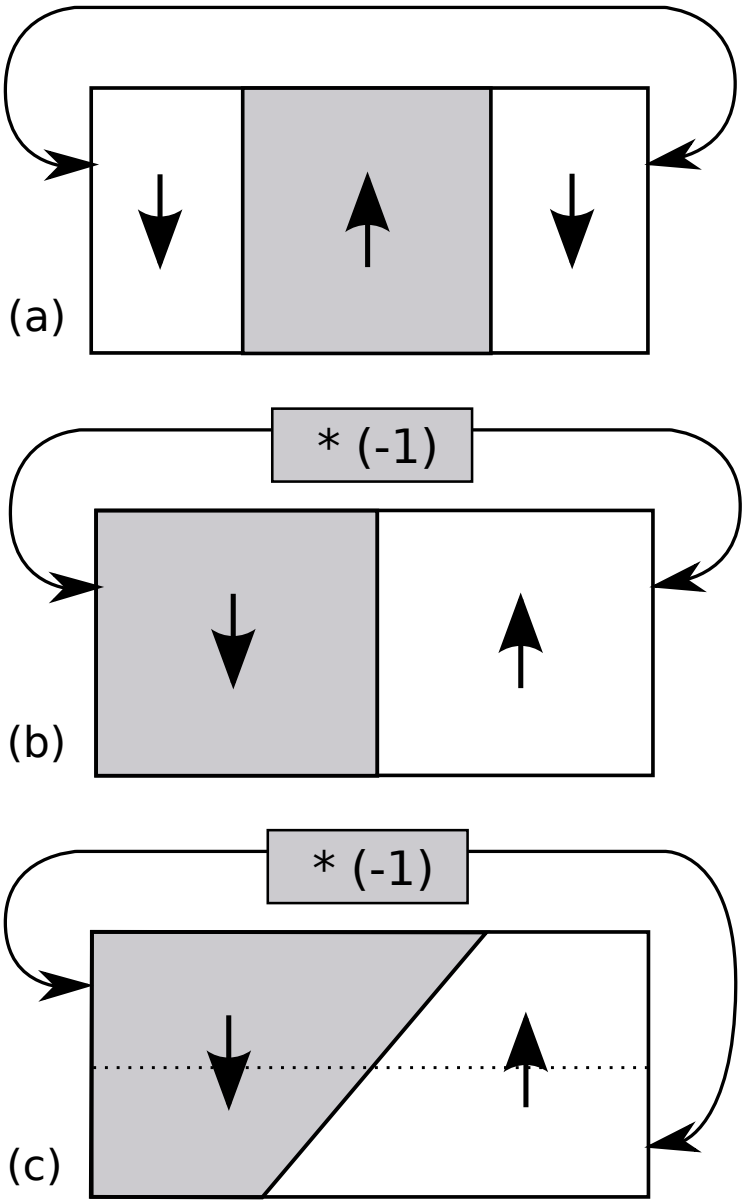


Figure 2.1: Periodic Boundary Conditions (a) can stabilize a *slab* geometry with two flat interfaces. This is only possible if the number of up/down spins is preserved. In a grand canonical configuration, the system will demix into one of the two phases. A canonical Ising simulation scheme is presented in chapter 3, though it is not used in this thesis. With antiperiodic boundary conditions (b), a single flat interface can be stabilized even in the *grandcanonical* ensemble. This is beneficial for our problem, since a *grandcanonical* simulation of the Ising system is a lot easier to parallelize and to run on a GPU. With a nonzero wall field, the mirror symmetry by reflecting the x-axis is broken, so we have to introduce another kind of boundary condition (c) in the y-direction to reflect the new symmetry properties of the system.

2.2.2 Periodic Boundary Conditions (PBC)

A common choice for boundary conditions are *Periodic Boundary Conditions* (PBC) which means that the opposite ends in each dimension of the simulation domain are connected, and interaction energy is transmitted over the boundaries of the system. The borders in each dimension are connected to each other (figure 2.1 top). This has the topology of a (d + 1) dimensional torus where d is the dimensionality of the simulation box. In the lattice gas interpretation, a particle that leaves the box on one side will enter the box again on the other side, and a particle on the edge of the system will feel the interaction of a particle on the other side of the box. By choosing the boundary conditions to be periodic, we can eliminate the effect of the presence of hard walls, but we will not eliminate the effect of the finiteness of the system altogether. The results we will get in this system will be influenced by *finite-size effects*. We choose the periodic boundary conditions to be transmitted at the walls in y-direction, orthogonal to the hard walls.

With periodic boundary conditions, an interface can only be stabilized under a *canonical* constraint, leading to a *slab* geometry (see figure 2.1 top). In the *grandcanonical* ensemble, statistical fluctuation eventually will take the system to a one-phase state where the interfaces have disappeared.

2.2.3 Antiperiodic Boundary Conditions (APBC)

To stabilize a flat surface in our simulation domain, we introduce the concept of *Antiperiodic Boundary Conditions* (APBC). This can only be defined in systems featuring a symmetry between the coexisting phases like the Ising model, and means that instead of transmitting the interactions from one end to the other end of the system, the exact opposite configuration is transmitted. When the neighboring spin on one end of the system is up ($\sigma_{L_y-1} = 1$), for spin σ_0 on the other side of the system, the image of σ_{L_y-1} that gets summed up in the nearest neighbor sum of σ_0 is actually -1. A similar surface-stabilizing effect could be achieved with hard walls with opposite interaction energy on either side of the simulation domain.

In the next section we introduce a slightly modified kind of boundary conditions, which addresses a problem that became apparent only gradually in our simulations (see figure 2.2).

2.2.4 Generalized Antiperiodic Boundary Conditions (GAPBC)

As can be understood from figure 2.2, we have to be careful with the antiperiodic boundary conditions. If the system exhibits a single interface, it is translationally invariant in y-direction in the sense that translation of the interface in

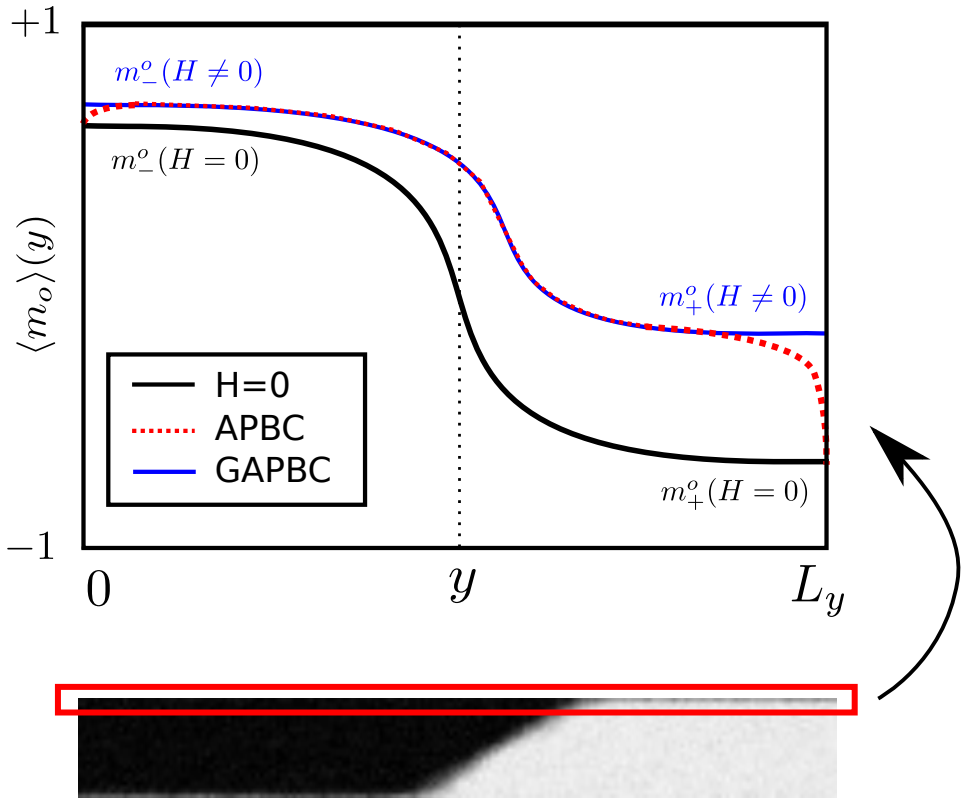


Figure 2.2: The profiles of the average magnetization in the uppermost layer $\langle m_o \rangle(y)$ as a profile along the y-axis reveal a problem that needs to be addressed. At $H = 0$ (black line) the profile is perfectly antisymmetric ($m^o_- = -m^o_+$), so antiperiodic boundary conditions (section 2.2.3) work, since the magnetizations in the bulks are nearly constant even near the system border (which is an important property for later measurements). When $H \neq 0$ however, the symmetry is broken, which leads to border artifacts for antiperiodic boundary conditions (red dotted line). The GAPBC boundary conditions introduced in section 2.2.4 solve this problem.

y-direction does not cost any free energy. For $H = 0$, APBC reflect the symmetry of the system

$$\begin{aligned} m^{(+)}(x,0) &= m^{(+)}(L_x - x, 0), \\ m^{(-)}(x,0) &= m^{(-)}(L_x - x, 0). \end{aligned} \tag{2.6}$$

For $H \neq 0$, this symmetry is broken. The asymmetry of the magnetization profile near the borders in y-direction (figure 2.2, red dotted line) leads to an extra contribution to the free energy of the system and thus a systematic error in our

calculations. If $L_y \gg L_x / \tan\Theta$, the boundary conditions do not disturb the interface and the effect illustrated in figure 2.2 *may* be negligible. For small enough system sizes L_y or for large enough angles Θ, they are not. With antisymmetric surface fields, the magnetization profile in x-direction has a different symmetry property

$$m^{(+)}(x, H) = -m^{(-)}(L_x - x, H) \tag{2.7}$$

Here the upper subscripts $(+), (-)$ distinguish the two phases with positive or negative total magnetization that may coexist with each other at zero bulk field. Changing the sign of the magnetization (both in the bulk and locally) together with the spin of the surface field leaves the profile invariant only together with the reflection at the midplane $L_x/2$. The generalized antiperiodic boundary condition must respect this antisymmetry property of the profiles, i.e.

$$S(x, y, z) = -S(L_x - x, y \pm L_y, z) \tag{2.8}$$

This is written here in a continuum interpretation of coordinates, but can be generalized immediately to a discrete lattice, then the coordinates in x-direction are labeled as $0, 2, ..., L_x - 2, L_x - 1$ and hence

$$\begin{aligned} S(0, y, z) &= -S(L_x - 1, y \pm L_y, z) \\ S(1, y, z) &= -S(L_x - 2, y \pm L_y, z) \\ S(2, y, z) &= -S(L_x - 3, y \pm L_y, z) \\ &... \end{aligned} \tag{2.9}$$

Thus the proper description of the boundary condition is an antiperiodic boundary condition with a Moebius strip topology in one dimension. Since the other two dimensions have a torus topology, the whole construction is a three dimensional generalization of a Klein bottle. A finite-size scaling analysis has been carried out for an Ising system with this topology in [59]. GAPBC perfectly reflect the geometry of the model system, thus removing the border artifacts altogether (figure 2.2, blue line). Also, there is no need to fix the interface in the center.

The GPU implementation of the GAPBC was less convenient. Unfortunately the spin layout chosen in the GPU implementation was developed before the spacial symmetry problem became apparent and therefore was not optimal for this purpose (section 3.3.7). Nevertheless, we managed to effectively hide the computational overhead.

2.3 Simulations on GPUs

For the investigations we are aiming to carry out we need very large simulation domains. The geometry with the flat interface is only useful if the two phases are large enough to be considered *bulk* phases and *finite-size effects* are controllable. A typical system can be seen in figure 2.3. To do this efficiently, we have to accept a few restrictions to our simulation method. Since we want to benefit from the additional computational power of GPUs, we need to update spins in parallel. We are not allowed to update spins in parallel which are interacting with each other. A pattern that works well is the checkerboard update (figure 2.4). The actual implementation and the technical complications that come with it are detailed in the next chapter. The second restriction is that we can only perform single spin flips efficiently, thus it is not possible to keep the total number of spins constant during the simulation. This is why we chose to simulate the system in the *grandcanonical* ensemble, which means we keep the temperature T constant, but we neither impose a constraint on the total number of up/down-spins, nor on the total energy of the system. How a simulation with constant particle number can be implemented is suggested in section 3.6, it is not used for any practical simulations in this thesis though.

Using (generalized) antiperiodic boundary conditions, an interface can be stabilized even in the *grandcanonical* ensemble. The system is initialized at Monte Carlo time $t = 0$ with the left side of spins all pointing downwards and the right half of the spins all pointing upwards (see upper left configuration of figure 2.5). Due to the choice of ensemble, the interface is performing a random walk. This has been verified by measurement as shown in figure 2.6. After initialization the system is expected to equilibrate after a few Monte Carlo steps into a stable physical configuration with an interface in the middle.

Each surface has a spectrum of capillary fluctuations. Initially the surface is perfectly flat, and the amplitudes of all capillary waves are zero. Equilibration is complete after they have fully developed, and long wavelength capillary waves take longest to equilibrate (which can be compared to critical slowing down). In thermal equilibrium, the mean square amplitudes of the capillary waves are given by the equipartition theorem. We gather statistics from the system after the system has been equilibrated. Since the interface performs a random walk, it will eventually hit one of the system borders in y-direction. To prevent side effects by the influence of the system borders and to avoid complications in the measurement of the coexistence magnetizations $\langle m^o_- \rangle$ and $\langle m^o_+ \rangle$ (see figure 2.2), we reset the configuration after 100.000 steps to the initial configuration at time $t = 0$, equilibrate again and gather statistics again.

In typical simulation runs, we used an equilibration period of 10.000 full lattice

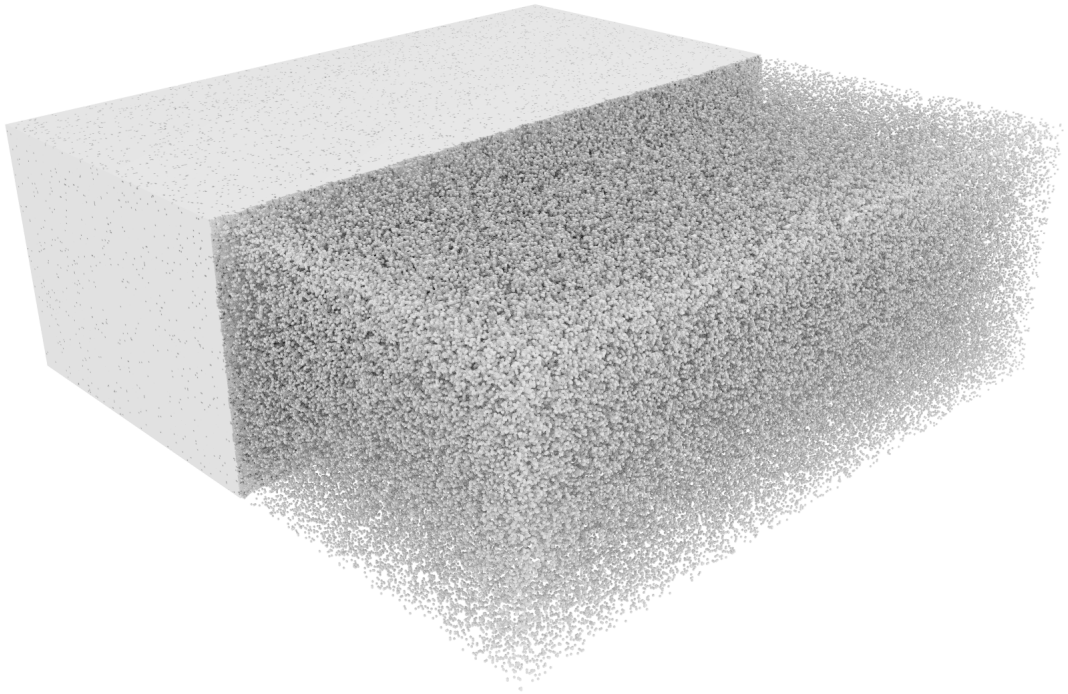


Figure 2.3: A full configuration of size $184 \times 504 \times 504$ at $T = 2.0$. This shows the scale of the simulations performed in this thesis.

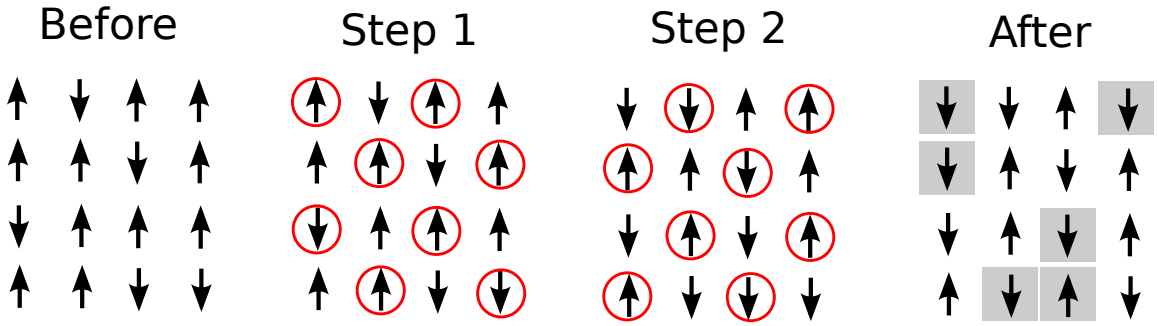


Figure 2.4: The checkerboard spin update procedure. Spins are selected and updated in two steps. In the first step, only spins from the white sites of a checkerboard are updated, while in the second step, the remaining spins are updated. After the two steps, each of the spins in the lattice has been updated once.

updates and collected statistics until we performed a reset to the initial state after 100.000 full updates. We have ensured that the systems are equilibrated after 10.000 steps and that they did hit one of the borders after the production period.

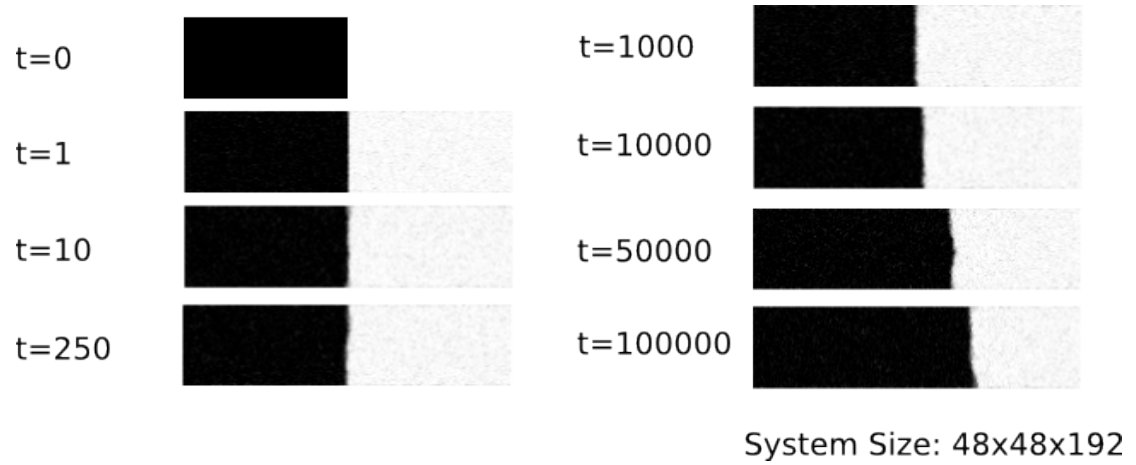


Figure 2.5: The density profile in the $x - y$ plane at different Monte Carlo times t. In this specific case, notice the drift of the surface in the middle of the system to the right side. This is not a systematic drift, but the surface performs a random walk. Nevertheless, it is expected to eventually hit one of the borders of the system.

During the last three years we have employed a variety of GPUs from different generations and vendors i.e. NVidia Geforce GTX 480, NVidia Geforce GTX 580, NVidia Geforce GTX 690, NVidia Geforce TITAN, NVidia Tesla K20X and a AMD Radeon HD 6970 (OpenCL). The details of the implementation on the GPUs are discussed in the next chapter.

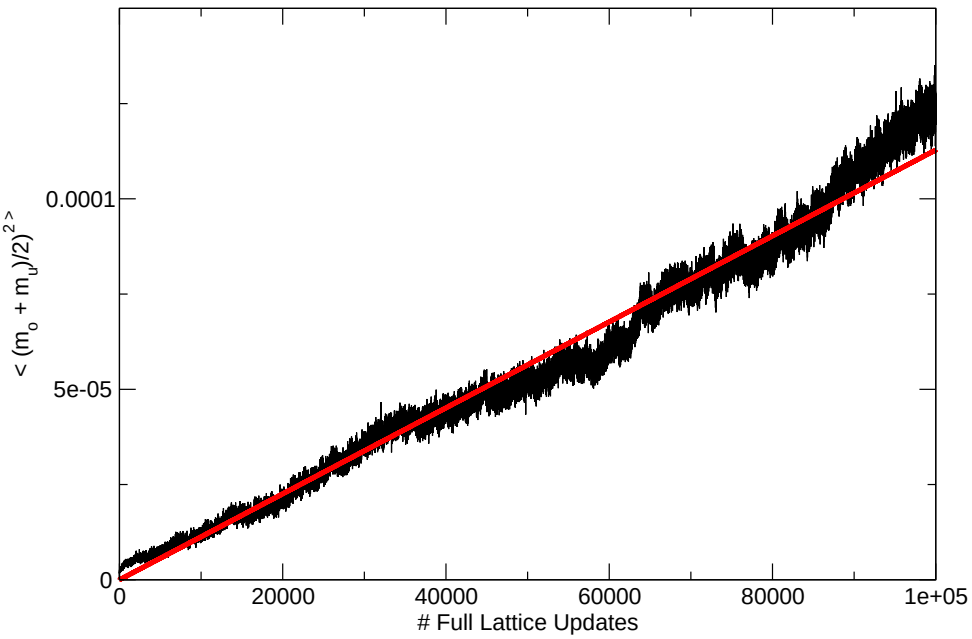


Figure 2.6: The average of the magnetizations in the uppermost (m_o) and the lowermost (m_u) layer can be easily extracted from a GPU simulation. This data has been extracted from a 10^5 full lattice update run of a $88 \times 504 \times 504$ system. The data has been averaged over about 20 runs. The result is proportional to the mean square displacement of the interface which is expected to perform a random walk. The end to end mean square displacement of a random walk is expected to be a linear function of the amount of steps. A linear dependence can be verified in our data (red line).

Chapter 3

GPU Implementation of the Ising Model

3.1 Introduction

The Monte Carlo method and computational methods in general have had a significant impact on traditional sciences throughout the last decades. Physical and engineering obstacles in microprocessor design have resulted in an ever decreasing growth of non-parallel microprocessor performance in the struggle to keep up with the ever growing demand of computing power. This is why we have seen a general trend towards multi-core and massively parallel computing architectures.

Graphics Cards are dedicated devices which were initially designed to accelerate the common tasks in 3D graphics in hardware. Initially they were designed to accelerate high end graphics workstations such as SGI's Onyx Reality Engine machine, which introduced the OpenGL standard in 1992. They evolved into the consumer market, ultimately with mass products such as 3Dfx's Voodoo graphics accelerators for personal computers and SGI's RCP chip powering the Nintendo 64 around 1996, which eventually caused a revolution in terms of graphics quality in video games. The tasks performed in computer graphics tend to be of a parallel nature, such as *transform each point in a given geometry according to the procedure vs()* or *perform the same operation ps() on each pixel covered by a triangle*, so the hardware implementing those operations had to be of a massively parallel nature, offering many parallel cores which could do the same operation on a number of data sets in parallel. At first, the hardware implementing the operations vs() and ps() were offering a tunable, but fixed functionality, which lead to a certain uniformity in the looks of computer graphics and computer games. After GPUs grew more mature, the industry demanded more flexibility, thus leading to the idea of programmable *Graphics Processing Units* (GPUs). The operations vs() and ps() where called *vertex* and *fragment* shaders, following the concept of the programmable *shaders* in Pixar's RenderMan [60] package, and could be freely programmed, first in a machine language that was native to a specific GPU and later in higher level languages such as GLSL or Cg. Recent computer architectures are following the role-model example of GPUs, and future hard-

ware architectures are expected to implement a similarly parallel approach to computing. With new, more flexible programming interfaces GPUs can be utilized already today to perform general purpose computing in fields much different from computer graphics. Competing standards and solutions for solving portability issues and high level abstractions exist. GPUs are high-performance many-core processors that can be used to accelerate a wide range of applications. Significant speedup factors in comparison to conventional CPU implementations have been reported in a large variety of fields: Molecular Dynamics simulations as in [61, 62, 63, 64, 65], lattice Boltzmann simulations [66, 67], time series analysis focused on financial markets [68], radiative transfer models [69], modeling Tsunami waves [70], neural networks [71], protein sequence alignment [72] and many more.

This part of the thesis is loosely based on three publications, namely [73, 74] and my original implementation of the 2D Ising model which was implemented as part of my diploma thesis [75] and published in [76]. This in turn was motivated by the pioneering work of Preis et al. [68]. The parts of this chapter are at times almost identical to these publications with the exception of this introduction, additional details in the descriptions where they are helpful and the description of a canonical simulation. This chapter does not only provide a documentation of my implementation but is also a general review of work done in the field of Ising simulations on GPUs, so not all the methods presented have been implemented by me.

3.2 GPGPU History and CUDA

Usability of GPU computing APIs (Application Programming Interfaces) and portability of GPU programs have seen a steady improvement. Early programmable GPUs were programmed in an assembly style language which was specific to one single device architecture. Graphics Specific APIs addressed this issue by developing high level languages such as Cg (Nvidia), GLSL (Khronos Group) and HLSL (Microsoft). A survey of GPGPU on these earlier platforms can be found in [77]. There also exists pioneering work on the Ising model from this time [78].

With the introduction of the Compute Unified Device Architecture (CUDA) in 2007, Nvidia addressed the fact that GPUs were now used not only for graphics applications but for more general purposes in a wide range of fields [79]. Up to date, most simulation codes are written specifically for Nvidia devices, especially the CUDA API, which makes them, even though they are written in a C style language, very dependent on this specific architecture. There are several competing standards to achieve portability, and high level frameworks and libraries [80, 81] as well as attempts to automate CPU-GPU communication [82]

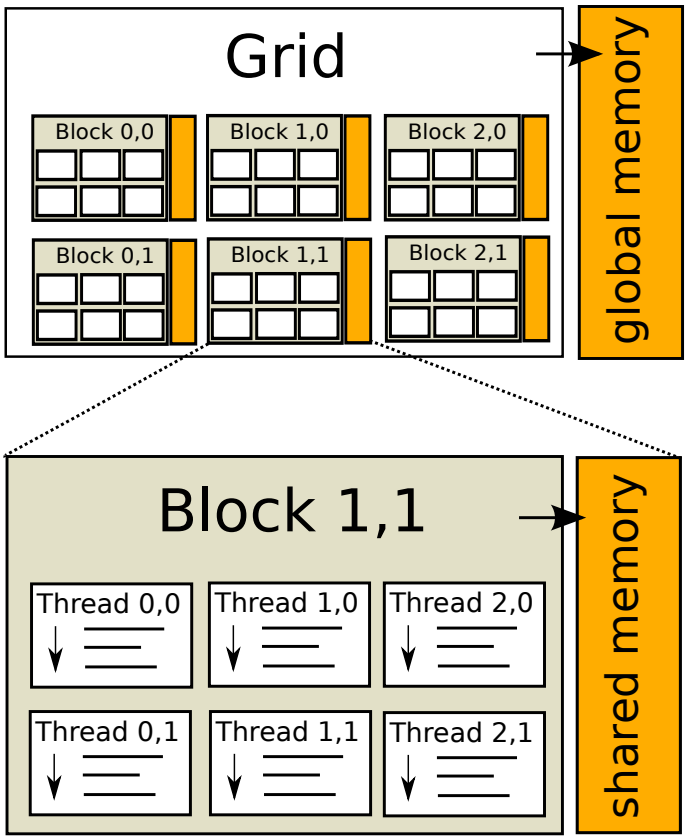


Figure 3.1: Hierarchical parallel execution of *kernels* on an Nvidia GPU. *Kernels* are executed by parallel threads, which are organized in blocks, where the whole block has access to a small shared memory space which can be used to accelerate block local computations. The blocks are arranged in a grid. Each thread in the whole grid can access a slower kind of memory, the global memory. There is also a memory that is local to each thread, and can not be shared between threads at all. Synchronization between threads is possible on the block level, but not on a global level.

and to achieve source code level portability between CPU and GPU as in [83] as well as a framework for porting shared memory GPU applications to multiple GPUs [84]. The purpose of this chapter is to show that a simulation can be designed in a way that it maintains portability across many platforms without sacrificing high performance and efficiency. The main APIs presented and used in this chapter are CUDA and OpenCL (Open Compute Library). A comprehensive quantitative performance comparison was done in [85].

CUDA is a very mature standard for General Purpose GPU computing. It is vendor specific to Nvidia and thus only works on Nvidia cards, without third party tools. GPU Ocelot exists as a framework to overcome this limitation and offers a way to execute and debug CUDA *kernels* on arbitrary hardware [86]. OpenCL however, is an industry wide standard, a unified interface for parallel computing, be it on GPUs, on multi-core CPUs or other future parallel platforms. The most important supporters of this standard are Intel, Advanced Micro Devices, Nvidia, and ARM Holdings. Both APIs allow programmers to write parts of their programs as small procedures that can be executed in parallel (so called *kernels*) on the underlying parallel acceleration hardware. These programs can

be specified in a C like language. CUDA gradually starts to support more and more C++ features, like templates and inheritance.

There are some major differences between the APIs. The most important one is that CUDA code needs to be compiled to byte code before the program is started, while OpenCL allows on-the-fly compilation from source. This is the only difference that cannot be fully hidden in the presented abstraction. CUDA consists of two different APIs (Driver API and Runtime API) which serve different purposes. The Runtime API is meant to allow easy access to the CUDA functionality and allows easy integration of CUDA into existing programs. The Driver API is the underlying API that is used by the Runtime API, but can also be accessed directly. In our implementation, we use the Driver API for CUDA, since it behaves more like OpenCL and we can find a unified way to hand over data to *kernels* and execute them.

A program that can be run in parallel on an Nvidia GPU in CUDA is called a *kernel*. *Kernels* are defined as functions in a language that resembles the C programming language. A *kernel* can be executed in multiple equally-shaped thread blocks, so that the total number of threads is equal to the number of threads per block times the number of blocks. Blocks are organized into a one-dimensional, two-dimensional, or three-dimensional grid of thread blocks as illustrated by figure 3.1. The number of thread blocks in a grid is usually dictated by the size of the data being processed or the number of processors in the system, which it can greatly exceed. This is described in detail in [87]. There are different kinds of memories with varying access speeds and policies that can be used to store data during the execution of a *kernel* (figure 3.1). The slowest memory is the global memory, which is shared between all threads. The fast shared memory is very limited in size and is shared between threads in each block. Last, there is the fast local per-thread memory which is used to store values within one thread and cannot be shared between threads. Using this memory hierarchy effectively is the main challenge in writing GPU optimized parallel code. OpenCL uses slightly different nomenclature, although the basic concepts are the same.

3.3 The Ising Model on the GPU

Recall that the interaction of the spins in the Ising model is given by the Hamiltonian

$$\mathscr{H} = -J \sum_{\langle i,j \rangle} S_i S_j - H \sum S_i \tag{3.1}$$

where the exchange constant $J = 1$ in the following and H denotes an external magnetic field. The lattice is updated according to the Metropolis criterion (see

section 2.1). As we have explained in the same wording in [73], the probability $W(t_i)$ for a Monte Carlo step from Monte Carlo time t_i to Monte Carlo time t_{i+1} to be accepted depends on the energy difference $\Delta\mathscr{H} = \mathscr{H}(t_{i+1}) - \mathscr{H}(t_i)$ between those states and is given by $\exp(-\beta\Delta\mathscr{H})$.

If no external field H is present, only discrete values for $\Delta\mathscr{H}$ are possible (two non-trivial values in the two dimensional case and three values in the three dimensional case). So if $H = 0$ the exponential factor can be pre-calculated on the CPU for each temperature and transferred to the GPU when the *kernels* are invoked as described in [76]. In this current implementation we go a step further and assume $H \neq 0$. This means we can make efficient use of the GPU in calculating the exponential factors for arbitrary fields H.

3.3.1 Checkerboard Update

In this section, where parts have been published word for word in [74] and other parts in [73], we will develop a very simple yet already efficient update scheme for the 2D Ising model on the GPU to illustrate the general idea. Parallel updates (i.e., parallel spin flips) can only be done for non-interacting subsystems. The fact that each spin only interacts with its four nearest neighbors makes a checkerboard update feasible.

To make efficient use of the GPU device structure, a parallelizable spin-update scheme has to be utilized. The ratio between memory latency and processing time on graphics cards is very large [39]. Thus, GPU cores can perform hundreds of instructions in the time of a single access to global memory. By highly parallel processing, memory access latencies can be hidden effectively, and large acceleration factors can be achieved. Parallel spin updates of the Ising model can only be done for non-interacting domains. The approach that each spin only interacts with its four nearest neighbors makes a checkerboard update feasible [22]. The lattice update is divided into two update steps A and B. In step A, only the spins residing on a black site are updated since they are not interacting with each other. In step B, the spins on white lattice sites are updated. It is essential that update step B is started after all updates of step A are finished. Please note, that other methods for the spin updating process are also available, e.g. diverse cluster algorithms [44,45], which perform particularly well close to the critical point. However, the systematic scheme of the checkerboard algorithm is most suitable for the GPU architecture realizing non-interacting domains where the Monte Carlo moves are performed in parallel. In the checkerboard parallelization scheme, one half of the spins does not interact with the other half of the spins in one timestep and can be updated in parallel. Early work ([68]) provided a feasibility study on how to implement the Ising model on Nvidia's CUDA platform, while later studies focused on maximizing the efficiency of the

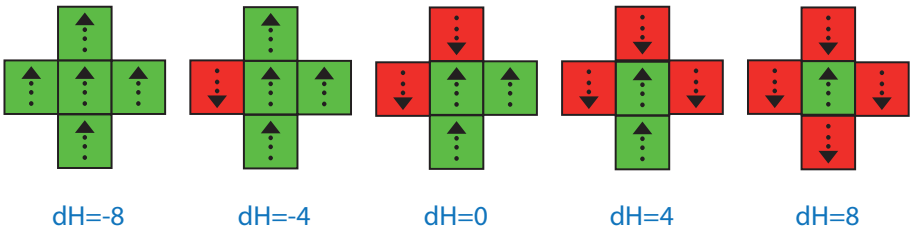


Figure 3.2: At each time step a spin is flipped from up to down or from down to up in dependence on the local interaction energy difference before and after the flip. There are 5 possible energy differences in the two dimensional case. Negative energy differences and energy differences of value 0 are always accepted, so only two exponential factors need to be calculated. This can be done in advance to save computation time. The image was taken from [74].

implementation [88, 89, 76] and spreading the simulation across many GPUs [90, 76, 91]. Recently, even cluster algorithms have been implemented ([92, 93]), even though speedups are somewhat less impressive. The lattice is updated according to the Metropolis criterion [57] for each step. For each step, the energy difference $\Delta\mathscr{H} = \mathscr{H}_a - \mathscr{H}_b$ between two subsequent states a and b is calculated. The probability for the step to be accepted is given by $W_{a\to b} = \exp(-\Delta\mathscr{H}/k_B T)$ if $\Delta\mathscr{H} > 0$ and $W_{a\to b} = 1$ if $\Delta\mathscr{H} \leq 0$. Since only discrete values for this factor are possible they should be pre-calculated on the CPU for each temperature and copied to the GPU when the *kernels* are invoked.
Detailed balance is only fulfilled if spins are selected randomly from the spin lattice and if the updates are done in succession, not in parallel. Since on the GPU, parallel updates are needed for an efficient implementation, detailed balance has to be sacrificed and a checkerboard update is used instead which can only fulfill the weaker condition of *global balance*.

3.3.2 Simple Scheme in 2D

In this section we present a very simple example implementation that has been published in the same wording in [74]. The lattice update is divided into two update steps (a) and (b). In step (a), only spins residing on a black site of a checkerboard structure are updated since they are not interacting with each other. In step (b), the spins on the white lattice sites are updated. For clarity, we use a data structure *ddata*, which holds all the pointers to data stored on the GPU, and a data structure *hdata* which holds all pointers to data in the host memory. We will now provide a simple implementation of the Ising model in plain CUDA (as it has been published in the mini-review [74]). Here, we store an array of integer variables in the global memory, which represents the two

dimensional spin lattice of linear size N_{LINEAR}—the total size of the array thus has to be $N_{\text{QUAD}} = N_{\text{LINEAR}} \cdot N_{\text{LINEAR}}$.

```
cudaMalloc((void**) &ddata->spins,sizeof(int)*N_QUAD);
hdata->metaSpins = new int[N_QUAD];
```

For each parallel thread we will need a random number generator (see section 3.3.6) which needs a seed that has to be stored in global memory too. So we make sure there is memory allocated for these seeds:

```
cudaMalloc((void**) &ddata->randomNumbers,sizeof(int)* NUM_THREADS);
hdata->randomNumbers = new unsigned int[NUM_THREADS]
```

A grid layout has to be designed in a way that allows parallel threads to update different lattice sites with a tunable block size (see Fig. 3.3). In this simple layout with a total of $(N_{\text{LINEAR}}/2) \cdot (N_{\text{LINEAR}}/2)$ threads, each thread processes a field of 2×2 spins, where only two of the spins are processed in each update step (the black sites in update step (a), the white sites in update step (b)). The threads are grouped into blocks of size BLOCK_DIM. The positions of the spins to be updated can be constructed from the block and thread indices as follows:

```
int tx = (threadIdx.x + blockIdx.x * BLOCK_DIM) * 2;
int ty = (threadIdx.y + (blockIdx.y % FIELD_DIM) * BLOCK_DIM) * 2;
```

These indices address the upper left spin for each thread. The spin field can be accessed, e.g., via the macro

```
#define _F(_x, _y) ((_x) + N_LINEAR * (_y))
```

For each update, the energy difference for a spin to be flipped from -1 to 1 or from 1 to -1 has to be calculated according to Fig. 3.2. For the upper left spin, the energy difference is

```
dH = 2 * spins[_F(tx,ty)] * (
    spins[_F(tx+1, ty)] + spins[_F(tx-1, ty)] +
    spins[_F(tx, ty+1)] + spins[_F(tx,ty-1]
    );
```

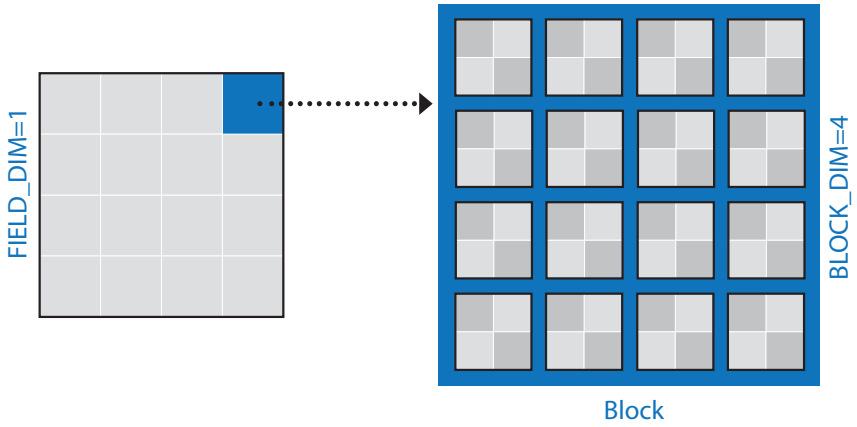


Figure 3.3: Checkerboard layout for parallel thread updates. The image was taken from [74].

Here, one needs to take care of the boundary conditions: If a memory access leaves the boundaries of the memory layout, it has to be wrapped so that it accesses the memory entry on the opposite site of the system to realize periodic boundary conditions. Using the energy difference dH, the probability for a spin to be flipped can be calculated using an exponential function. With a previously drawn random number *ran* between 0 and RAND_MAX, the spin flip is simple to implement:

```
if (ran < exp(-dH / t) * RAND_MAX)
    spins[_F(tx,ty)] = -spins[_F(tx,ty)];
```

This code segment updates the spin lattice using a write to global memory. Exponential factors should be calculated in advance, since there are only two relevant energy differences ($dH = 4$ and $dH = 8$). All other energy differences lead to spin flips anyway.

This implementation is not as efficient as possible, but it served for the explanation of the algorithm. An optimization of this update scheme will be discussed in section 3.3.7.

3.3.3 External Fields

External fields can be incorporated into the simulation by providing the external field either in an analytical form that can be evaluated directly on the GPU, or in terms of an array in global memory that assigns each lattice site a value of the external field. Problems arise when using more sophisticated schemes as in [76, 94], especially when the efficiency of the simulation relies on pre-calculated

exponential factors in some form, because the external field modifies the value of the exponential factor that is used to evaluate the Metropolis criterion.

3.3.4 Extraction of Magnetization

After each Monte Carlo step, we have to extract the magnetization of the spin lattice. The magnetization is given by

$$m = \sum_i s_i. \tag{3.2}$$

The idea for fast summation is to use a binary tree reduction for each block. Each thread calculates the sum of all its spins (2×2), and writes it into an array in shared memory. After each thread has written into shared memory, a binary tree reduction[1] can take place:

```
/* Save partial results back to shared memory in new structure */
__shared__ int r[BLOCK_DIM * BLOCK_DIM];
int id = threadIdx.x + BLOCK_DIM * threadIdx.y;
r[id] = dH;
__syncthreads();
/* Reduction on GPU */
for(unsigned int dx=1; dx < BLOCK_DIM * BLOCK_DIM; dx *= 2) {
   if (id % (2 * dx) == 0)
     r[id] += r[id + dx];
   __syncthreads();
}
if(threadIdx.x == 0)
  out[blockIdx.x + FIELD_DIM * blockIdx.y] = r[0];
```

After each reduction step, threads are synchronized to prevent race conditions. Next, each block writes out the sum of spins for its spins into global memory (out). The sum over all entries in the out array can be done on the CPU after the *kernel* execution ends, since it is not performance critical. This section has been published in [74].

3.3.5 Cluster Updates

For the investigation of the critical behavior of non-disordered models, cluster algorithms [95, 96] will perform much better than any optimized implementation of a local spin-flip algorithm already for medium system sizes. In this

[1] See also the reduction examples provided in the NVIDIA CUDA SDK.

thesis, cluster updates have neither been used to gather data, nor have they been implemented on the GPU as part of the thesis. They are mentioned for completeness, since the parallelization of more sophisticated update schemes is an evolving field where there is a lot of room for future work. Wolff [95] proposed a Monte Carlo algorithm in which only a single cluster is flipped at a time. The spin-update process can be outlined as follows:

- Loop:
    1. Choose a random seed spin S_i and flip it.
    2. For all neighbors j: If (S_j) is parallel with S_i, add S_j to the cluster with probability $p = 1 - e^{-2\beta}$.
    3. After all nearest neighbors j have been checked, look at each of the nearest neighbors k of site j. If S_k is parallel with S_j and is not a member of the cluster, add S_k to the cluster with probability $p = 1 - e^{-2\beta}$. Flip S_j.
    4. For all added spins S_k, repeat (3) until no more parallel spin pairs are found.

A sub-lattice decomposition cannot be used for parallelization in the case of a cluster algorithm. [97] proposed an parallelization of Wolff updates using OpenMP first. The idea is to use parallel computation for the newly added spins in step (3). The newly added spins form wave-fronts and are thus referred to as wave-front spins. Each thread is a assigned a wave-front spin in the grid. [92] has recently developed a GPU implementation of this scheme.

The situation that different spins try to incorporate the same spin to the cluster simultaneously needs to be avoided. This leads to the problem of global thread synchronization, which is intrinsicly absent in CUDA. This can be addressed by different techniques, one of them is presented in [92]. The actual number of threads was chosen to be a maximum of 1024. With proper use of shared memories, the technique is very effective for fast computation. Access to global memory is time consuming, so access to global memory also here has to be reduced. A linear congurential random number generator as in [68] is used for flipping spins according to the transition probability $p = 1 - e^{-2\beta}$.

The performance achieved by the GPU implementations is up to 7.9 times as fast as a CPU, which is already higher than the performance increase reported in [97] and an impressive result.

A multi-cluster spin flip algorithm was proposed by Swendsen and Wang [96]. This algorithm was recently ported to the GPU for the q-State Potts model by [89] and also by [93]. The Ising model as a special case (q=2) of the Potts model

performs well on the GPU, achieving speedups of about 12.4 compared to a current CPU core.

The spin-update of the Swendsen and Wang cluster algorithm for on a CPU can be formulated as follows [96]:

- Loop until all spins S_i are checked:
  1. Choose a spin S_i.
  2. For each of the nearest neighbors S_j: If $S_j = S_i$, generate bond between site i and j with probability $p = 1 - e^{-2\beta}$.
- Apply the Hoshen-Kopelman algorithm [98] to identify all clusters.
- For all clusters:
  1. Flip the spins S_i in the cluster with probability 1/2.

Since the bond generation and the spin flips are done independently on each site, these steps are well suited for parallel execution on GPU. The cluster labeling however is done on each site piece by piece sequentially and cannot be computed in parallel on GPU. This section has been published word for word in [74].

3.3.6 Random Number Generation

For every update thread, one or more random numbers are needed to evaluate the Metropolis Criterion for different exponential factors. This is the reason why an efficient method to create random numbers is needed. [99] gives a good overview of pseudo-random number generation in Monte Carlo simulations and [100] is dedicated to the solutions of this important problem on the GPU.

A simple example of a pseudo-random number generator is the Linear Congruential Random Number Generator (LCG) [101]. In a simple straight-forward implementation [68], a single random number generator provides the random numbers for every spin update thread j. A sequence of random numbers for the j-th thread $x_{i,j}$ (where $i \in \mathbb{N}$) is generated by the recurrence relation

$$x_{i+1,j} = (a \cdot x_{i,j} + c) \bmod m \tag{3.3}$$

where a, c and m are integer coefficients. An appropriate choice of these coefficients is responsible for the quality of the produced random numbers. We use $a = 1664525$ and $c = 1013904223$ as suggested, e.g., in [102]. Since by construction, results on a 32-bit architecture are truncated to the endmost 32 bits, the modulo operation m is set to 2^{32}. By normalizing ($y_{i,j} = \mathrm{abs}(x_{i,j}/2^{31})$) the LCG

can be used to generate random numbers $y_{i,j}$ in the interval $[0;1]$. For the GPU, an array of random numbers that provides a single random number seed for every spin update thread can be generated by the iteration

$$x_{0,j+1} = (16807 \cdot x_{0,j}) \bmod m \tag{3.4}$$

with $x_{0,0} = 1$.

When using a LCG, which has a rather short period $p = 2^{32}$ of the generators, unfortunately, most of the different sequences will have an overlap and, so already a few complete sweeps of a lattice will significantly exceed the period of the generator, and even more dramatically exceeding the value $\sqrt{p}$ considered to be safe when using LCGs. In [68] it was sufficient to determine the critical point of the two dimensional and three dimensional Ising model with high accuracy.

If more sophisticated pseudo-random number generation is needed, alternatives are available. [89] compares different implementations of simple pseudo-random number generators (LCG and Fibonacci) for spin model implementations. The HybridTaus generator presented in [103] uses the output of a LCG and combines it with the output of a Tausworthe generator [104]. The CUDA SDK provides an implementation of the Mersenne Twister generator. The sample program runs a set of 4096 Mersenne Twister generators in parallel that run in a thread layout of 32 blocks which consist of 128 threads. If the thread layout is adjusted it should be possible to use it as a drop-in replacement for our presented LCG. It has a period of about 2^{607}. A pseudo-random number generator on a GPU called MTGP is presented in [99], which is a variant of Mersenne Twister. The period of this method is 2^{11213}. Finally, there is CURAND, which is also available in the CUDA SDK with a period of 2^{192}.

This section has been published word for word in [74].

3.3.7 Optimizing Memory Lookup

In my diploma thesis [75] and in [76], we proposed a memory encoding scheme for the 2D Ising model to account for the long latencies for memory access on a GPU by avoiding memory accesses and substituting them by computations which are very cheap on the GPU. We will describe it here before continuing to an improved scheme in section 3.3.8. The encoding was measured to perform nearly 200 times faster on a single Tesla C1060 GPU when combined with a reasonable update scheme, compared to a single CPU core of the Nehalem architecture that runs a straight-forward non-parallel implementation (see section 3.5). From this result, one can draw the conclusion that architecture specific optimization is very important—on the GPU as well as on the CPU—especially if performance comparisons are drawn between those architectures. The spin field in a memory layout is memory optimal and reduces memory lookups. The

spin field on the graphics card is encoded in blocks of 4×4 spins (hereafter referred to as "meta-spins") which can be stored as the digits of one unsigned short integer (2 byte) and can be accessed by a single memory lookup.

Single spin values can be extracted from a meta-spin using the *conversion*

```
s[x,y] = ((metaspin & (1 << (y * 4 + x)) != 0) * 2 - 1)
```

which returns a value of either -1 or 1. This way, each spin uses exactly one bit of memory. The spin field is stored in global memory which is very expensive to access. The allocated global memory is smaller by a factor of 16, compared to an integer spin representation: To process the spin field on the GPU, the spin field is subdivided into quadratic sub-fields which can be processed by threads grouped into one block. Each thread of this block processes a meta-spin of 4×4 spins. At the beginning of a *kernel* it retrieves 5 meta-spins from the global memory—namely its own and its four neighboring meta-spins (see Fig. 3.4 left). This information is used to extract the information of the 4×4 spins the thread will update as well as the neighboring spins into a 6×6 integer array in shared memory which allows fast computation of the spin flips. At the end of the *kernel* execution the 4×4 meta-spins are updated with one single global memory write. Although the update scheme sounds hardly efficient, it dramatically reduces global memory access compared to the implementation presented before which results in very fast computation times on GPU hardware.

In [76], a maximum lattice size of 100.000×100.000 spins can be processed on an individual GPU with 4 GB of global memory, using the given memory in an optimal way. This approach encodes 1 bit per spin. Furthermore, it utilizes fast shared memory for local computations. The whole lattice can be updated on a Tesla C1060 in roughly three seconds.

[89] uses another approach to reduce time spent on memory lookups based on a double checkerboard decomposition. A second, coarse checkerboard overlays the update checkerboard, organizing the spin lattice in line with the hierarchic memory layout of the GPU. On a fine level, the checkerboard is decomposed into blocks of $T \times T$ spins. This size has to be chosen so that all $T \times T$ spins can be stored in shared memory for faster access. These $T \times T$ spin-blocks are then arranged in a coarse checkerboard pattern to fill the whole lattice. Each of the spin-blocks is processed by one GPU thread-block with $(T/2) \times (T/2)$ threads, which use shared memory for their common memory accesses within the block. The lattice update is again updated in two steps. First all the white sites of the checkerboard are updated, and afterwards all the black sites are updated. Each of the update steps is again divided into two sub-steps. In the first sub step all the white sites of the coarse spin-block pattern are processed, and in a second sub-step all the black sites of the spin-block-pattern are processed. For each of

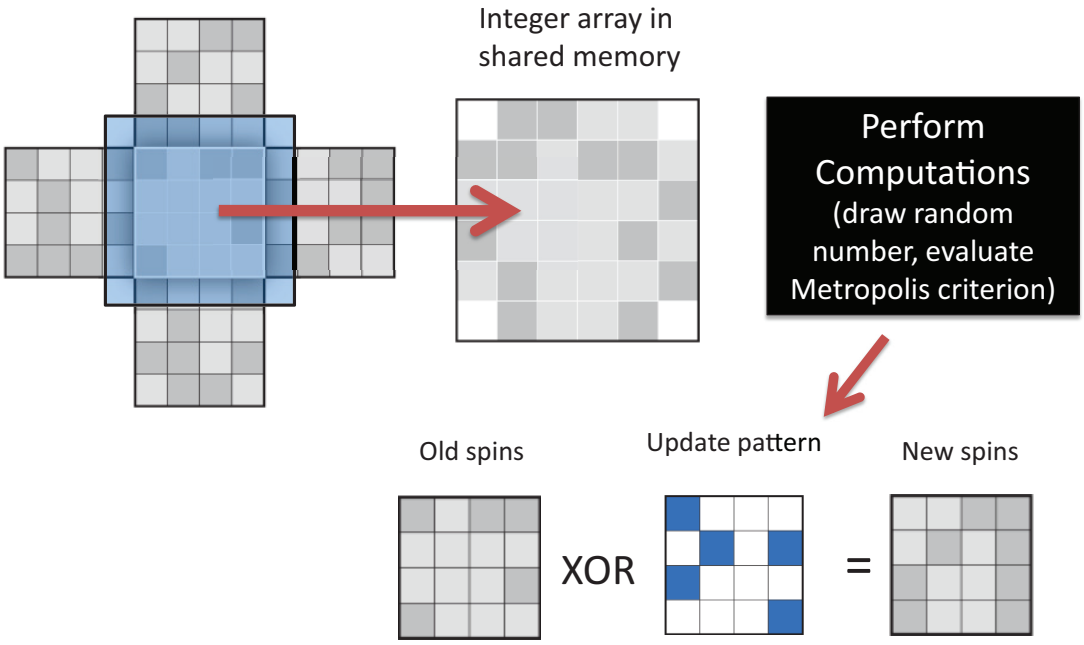


Figure 3.4: Procedure how each thread processes a 4x4 meta-spin. First, the meta-spin and its neighbors are looked up in global memory. After that, the actual spins are extracted into shared memory and an update pattern is created by evaluating the Metropolis criterion for each of the spins. After that, the new spins are obtained using the update pattern and written back to global memory. The figure was taken from [76].

the sub-steps, all threads have to extract the needed spins into shared memory, and computations can be done faster using only shared memory accesses. If more than one update of a spin block is done before memory is written back to global memory, special care has to be taken. For low temperatures the correlation length is typically smaller than the blocks of $T \times T$ spins. Close to the critical temperature however, method induced problems may arise as the correlation length exceeds the block size. This section has been published word for word in [74].

We did not use this memory layout in the present work, because this encoding is not reflecting the native memory access size of 128 bits of the NVidia achitecture, nor is it extendable to 3D. In section 3.3.8, we present a layout that works in

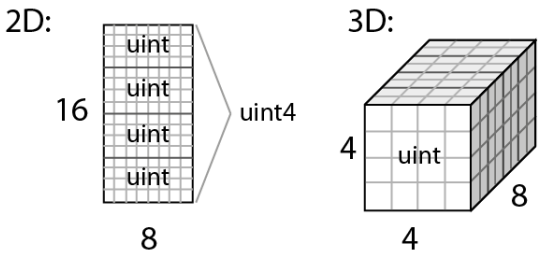


Figure 3.5: Organizing the spin lattice in 128 bit spin blocks makes memory lookup very efficient and the update logic fast. The figure was taken from [73].

2D and 3D, stores 128 spins into 128 bits and has been used for the computations in this thesis.

3.3.8 Revised Memory Layout and Update Scheme

This section has been published word for word in [73]. Global memory reads on a CUDA device are done in units of 128 bits. For our lookups to perform fastest, it is advisable to organize the simulation data in blocks of this minimum read size.

CUDA and OpenCL both provide intrinsic data types that come in sizes of 128bits, such as float4, int4 or uint4. We will use uint4 in our implementation, since each binary digit of the 4 32 bit integers can represent the state of a single spin (up or down).

For the purpose of explanation, we use the following definition in C++:

```
struct uint4
{
  uint32& operator[](uint32 i) { return e[i]; };
  union
  {
    struct { uint32 x, y, z, w; };
    uint32 e[4];
  };
};
```

It is important to find a memory layout for the simulation lattice, that makes efficient use of the memory structure of the underlying device. We present two memory layouts that organize the spin lattice into blocks of 128 spins, one for the two-dimensional and one for the three-dimensional case.

For the two dimensional model the blocks are organized in 8×16 spins, while in three dimensions they are organized in $4 \times 4 \times 8$ spins (see fig. 3.5 for a single

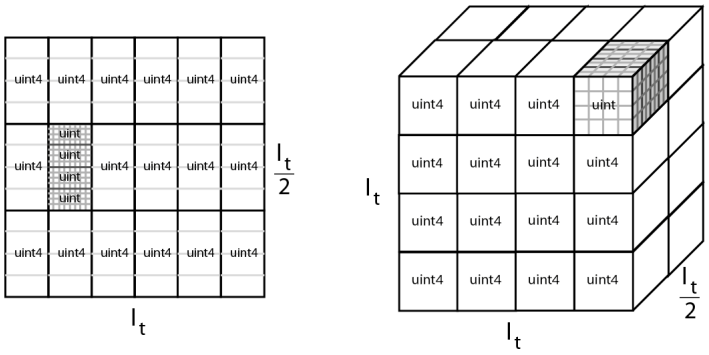


Figure 3.6: Organizing the spin lattice in 128 bit spin blocks makes memory lookup very efficient and the update logic fast. The figure was taken from [73].

block and 3.6 for the lattice layout). This is the size of one uint4 which is defined in CUDA as well as in OpenCL. For the host we can define it as follows: We define a size $l_b = 8$ for 2D and $l_b = 4$ for 3D, so the blocks are of sizes $l_b \times 2l_b$ for 2D and $l_b \times l_b \times 2l_b$ for 3D, and a size l_t so that the total spin field is of size $l_t l_b \times l_t l_b$ for 2D, and $l_t l_b \times l_t l_b \times l_t l_b$ for 3D. This way the spin field contains $l_t \times \frac{l_t}{2}$ blocks for 2D, and $l_t \times l_t \times \frac{l_t}{2}$ blocks for 3D. Each of this blocks is processed by a separate thread on the GPU.

A spin S_{ijk} at position (i, j, k) in the linear array for the spin field SF can then be looked up from memory by splitting up its index to:

$$(i, j, k) = (i_t \cdot l_b + i_b, j_t \cdot l_b + j_b, k_t \cdot l_b + i_b) \tag{3.5}$$

then first retrieving its spin block

$$SB(i_t, j_t, k_t) = SF[i_t + j_t \cdot t_t + k_t \cdot (l_t)^2] \tag{3.6}$$

and from that, the actual spin:

$$S(i, j, k) = SB(i_t, j_t, k_t)[i + j \cdot l_b + k \cdot l_b \cdot l_b] \tag{3.7}$$

For 2D, $k \equiv 0$.

Since this lookup has to be implemented using the native data type uint4, the actual lookup looks different because the 128 bits of data are spread over 4 32 bit unsigned integers. Also, single bits cannot be looked up directly but have to be extracted out of the 32 bit unsigned integers in memory. In the end, the bits need to be mapped to the spins they represent according to

$$[0, 1] \to [-1, 1] \tag{3.8}$$

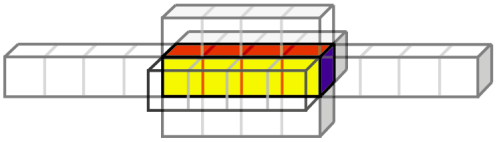


Figure 3.7: Extracting and processing a row of spins at a time. The figure was taken from [76].

So in reality, a memory lookup looks like

$$S(i,j,k) = SB_{\text{uint4}}[x \text{ div } 32]\&(1<<(x \bmod 32) \cdot 2 - 1 \tag{3.9}$$

with

$$x = i + j \cdot l_b + k \cdot l_b \cdot l_b \tag{3.10}$$

where mod denotes a modulo operation, div an integer division, & a bitwise and operation and << a bitwise left shift operation.

A spin block is most efficiently processed by extracting and processing one horizontal line of spins from a spinblock at a time. This works the same way in 2D and in 3D, with the difference that there are 4 neighboring spins to be looked up in 2D, and 6 neighboring spins in 3D.

A full line can be extracted from a spin block (block) (see figure 3.7) into a target buffer (targetBuffer) of ints (for fast computation) using the following routine:

```
extractLine(block, y, z, int* targetBuffer)
{
  int startPos = iBlock(0,y,z);
  int offset = startPos % 32;
  unsigned int spinBlock = block[startPos / 32];
  for (int i = 0; i < lb; ++i)
    targetBuffer[i] = ((spinBlock &
        ( 1 << (offset + i)))
        >> (offset + i))*2 - 1;
}
```

The extracted spins are used to evaluate the interaction energy difference in the Metropolis step, evaluate the Metropolis criterion and update the spins in the line accordingly. After the line is processed completely, the next line is processed by looking up and extracting the new required spins from memory, and keeping those in memory which can still be used from the previous line.

3.3.9 Multi-Spin Coding

This section has been published word for word in [74]. Multi-spin coding refers to all techniques that store and process multiple spins in one unit of computer memory. In CPU implementations, update schemes have been developed that allow to process more than one spin with a single operation [105, 106, 94]. In [76], we provide an approach with a scheme which encodes 32 spins into one 32-bit integer in a linear fashion. It was used as a CPU reference implementation since it performs very fast on the reference CPU. The 32-bit type is chosen since register operations of current hardware perform fastest on this data type. It depends on a large pool of 32-bit Boltzmann patterns, which are recalculated spin flip patterns that encode evaluations of the spin flip condition

$$r < \exp(-\Delta\mathscr{H}/k_B T)$$

for every single spin bit—the variate r is an independent and identically-distributed random number in $[0,1)$. Since for $H = 0$, there are only two possible energy differences $\Delta\mathscr{H}$ with $\Delta\mathscr{H} > 0$, two Boolean arrays can encode the information of an evaluation of the flip condition. For reasonable results, Ito [94] suggested to use a pool of 2^{22} to 2^{24} Boltzmann patterns. If we call the arrays exp4 and exp8, the encoding is chosen to store 0 (zero) into exp4 if $exp(8J/k_B T) < r < exp(4J/k_B T)$ and 1 (one) if not, and a 1 into exp8 if $r < exp(8J/kBT)$ and a 0 if not. [94] presents a way to encode the evaluations of these expressions in those pools of patterns in advance for each digit in a 32-bit pattern.

These schemes can perform extremely fast, however, they are very restricted since it is much harder to incorporate external forces. A GPU implementation was developed as well in the scope of [76], but because of the dependence on excessive random global memory lookups for precomputed random numbers, it proved not as fast as more straight forward implementation with GPU generated random numbers.

3.3.10 Multi-GPU Implementations

For a multi-GPU implementation, we can go a step further and split up the spin lattice into subdomains. These subdomains can be stored in the memory of different GPUs in different or the same machines and processed on those GPUs. This has been done as part of my diploma thesis [75] and published in [76] and has been proven to be very efficient.

For the simulation to remain physically correct, in each step we have to transfer the spins at the borders of the domain between neighboring GPUs (figure 3.8). In this current implementation, we go a step further and simulate one more

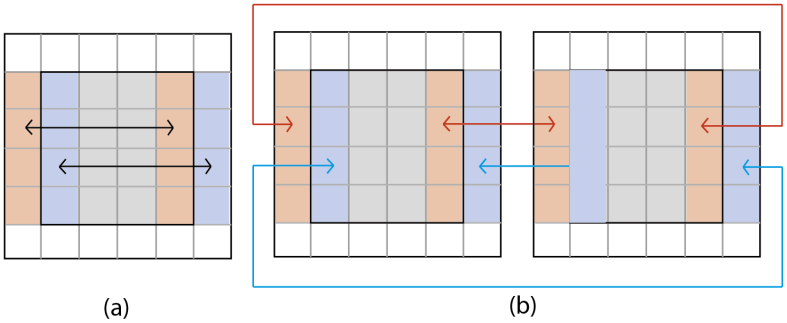


Figure 3.8: (a) shows how periodic boundary conditions can be implemented by simulating 1 excess row of spin blocks on both sides for each dimension. A memory exchange transfer is needed between the red and blue regions to propagate physical information across the border. This scheme can be used to spread the simulation lattice over more than one GPU for each dimension (b). Spin data has to be transferred between the domains as indicated by the red and blue arrows. The memory transfer is necessary after n_b time steps, where n_b is the linear dimension of the spin block, to make sure physical information is properly propagated over the domain borders. (Information spreads by one spin block per time-unit). The figure was taken from [76].

block at the border of each domain. This can even be used in the single GPU case, so periodic boundary conditions are simulated automatically without keeping track of them explicitly.

As shown in [76], higher performance can be achieved and larger system sizes are possible by putting multiple quadratic lattices next to each other in a super-lattice and let each lattice be handled by an array of multiple GPUs. When updating a single spin, it needs the information about its 4 neighbors. On the border of each lattice, at least one of the neighboring sites lies in the memory of another GPU. For this reason the spins at the borders of each lattice have to be transferred from one GPU to the GPU handling the adjacent lattice. This can be solved by introducing four neighbor arrays holding the spins of the lattices' own borders and four arrays for storing the spins of adjacent neighbors [76].

In [90], different, more sophisticated approaches for using multiple Graphics Processing Units in the simulation of spin systems are presented. The actual implementation benchmarks the 3D Heisenberg spin glass model, but it should be straight-forward to apply it to the Ising model. It is shown that it is possible to hide almost completely the communication overhead by using the CPU as a communication co-processor of the GPU. Large scale simulations on clusters of GPUs can be efficiently carried out by following the same approach also for other applications where a clear cut exists between bulk and boundary data.

3.4 Validation

The GPU implementations are usually compared to simple [68] or more sophisticated [76, 107] implementations on both CPU and GPU. As a measurement for the performance of an implementation, it makes sense to use the number of single spin flips per second, which also allows to compare results for different lattice sizes. The purpose of the CPU implementation usually is to have a fast and fair non-parallel reference implementation, not to benchmark the CPU. Therefore, most of the time, only one core of the CPU is used (without Hyper-Threading Technology).

The Binder cumulant U_4 is an extremely reliable test for the correctness of a Monte Carlo simulation for static simulations. It is defined as

$$U_4(T) = 1 - \frac{\langle M(T)^4 \rangle}{3\langle M(T)^2 \rangle^2}$$

where $M(T)$ is the magnetization per spin of a configuration for a simulation carried out at temperature T. It is used in [68] to prove the correctness of the parallel checkerboard update pattern, and in [92] to show the correctness of the parallel cluster updates using the Wolf algorithm.

Special care must be taken for all parallel spin update schemes when using them for dynamic studies as e.g. in [108, 109], because various investigations have shown that the different parallel update schemes lead to qualitatively different (to varying degrees) dynamics in Monte Carlo time. [89] uses the time correlation function to investigate the dynamic behavior of different versions of shared memory implementations. This section until this point has been published word for word in [74].

As mentioned before, the time correlation function of the checkerboard update, which selects the black and white sites alternatingly in a ABABAB pattern, shows that the time correlations decrease much more rapidly in a pure checkerboard update and thus lead to completely different dynamics. One thing we tried was to replace the ABABAB deterministic selection probability of the checkerboard sites with choosing black and white update steps randomly.

This leads to dynamics which are similar to a fully random selection, although the dynamics is unfortunately still distinct, and has even larger time correlations than a fully random selection.

3.5 Performance

We define the speedup of a GPU implementation in the same wording as in [110] as the ratio

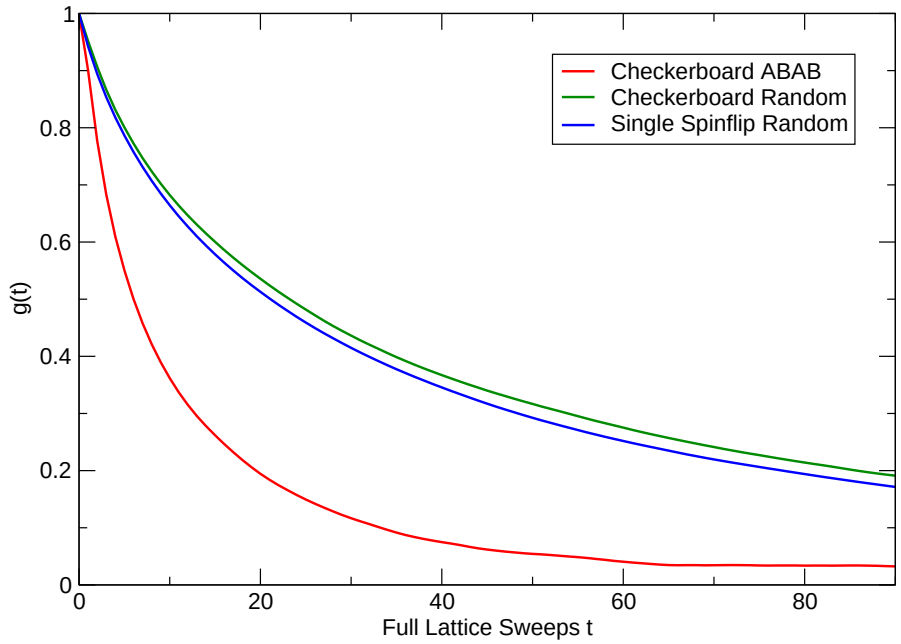


Figure 3.9: The time autocorrelation function $g(t)$ in a 2D Ising model of size 496×496. The pattern in which spins are selected for update have a significant effect on the spin correlations between different times. An interesting observation is that a deterministic ABAB checkerboard update leads to the fastest disappearance of correlations in the system.

$$s = \frac{t_{\text{CPU}}}{t_{\text{GPU}}} \tag{3.11}$$

between the time t_{CPU} that is spent on computation in a non-accelerated implementation and the time t_{GPU} that is spent on computations in a GPU accelerated implementation.

Since not all of the program code can be accelerated using a GPU, a fraction of the program code will always be executed on the CPU. Hence, we split the time t_{GPU} into two parts

$$t_{\text{GPU}} = t_{\text{GPU, accel}} + t_{\text{unaccel}}, \tag{3.12}$$

where $t_{\text{GPU, accel}}$ is the execution time for the part of the program that is actually

executed in parallel on a GPU and t_{unaccel} is for the rest of the code that stays in the CPU and is not executed in parallel.

For the CPU time, we can do the same split

$$t_{\text{CPU}} = t_{\text{CPU, accel}} + t_{\text{unaccel}}, \tag{3.13}$$

which yields the following formula for the speedup of the GPU accelerated program

$$s = \frac{t_{\text{CPU, accel}} + t_{\text{unaccel}}}{t_{\text{GPU, accel}} + t_{\text{unaccel}}}. \tag{3.14}$$

Here $t_{\text{CPU, accel}}$ refers to the execution time on CPU of that part of the program which we have also implemented in parallel in GPU accelerated version.

From (3.14), the maximum possible speedup corresponding to the case $t_{\text{GPU,accel}} = 0$ is

$$s_{\text{max}} = \frac{t_{\text{CPU, accel}}}{t_{\text{unaccel}}} + 1. \tag{3.15}$$

We benchmark the simulation package on an Intel Core i7 Nehalem machine with 2.67 GHz and a second to last generation consumer graphics card by Nvidia (Geforce 580) as well as on a comparable consumer graphics card by AMD (Radeon HD6970). As a measurement of the performance of an implementation, we use the number of single spin flips per second, which also allows to compare results for different lattice sizes. The temperature is set to $0.99 \cdot T_C$. Figure 3.10 and 3.11 show the quantitative results of the benchmarks for two and three dimensions. In figure 3.10, the different implementations of the two-dimensional system are benchmarked for different system sizes. The performance is roughly comparable with the speedups achieved previously in [76]. fig. 3.11 shows the same for the three-dimensional case. As we would expect, CUDA performance for the two dimensional Ising model is comparable to the results obtained in [76]. This shows that the framework could be generalized to different architectures and also to three dimensions without sacrificing efficiency.

3.6 Canonical Simulation

For a lot of problem-sets, it is more convenient to work in the canonical ensemble. The canonical ensemble can i.e. be used to stabilize a droplet [8, 111] phase, and investigate the line tension / surface tension of the droplet and compare it quantitatively to the results available and obtained for the flat surfaces.

A canonical simulation poses more problems for a parallel simulation, especially since the simulation needs to fulfill ergodicity.

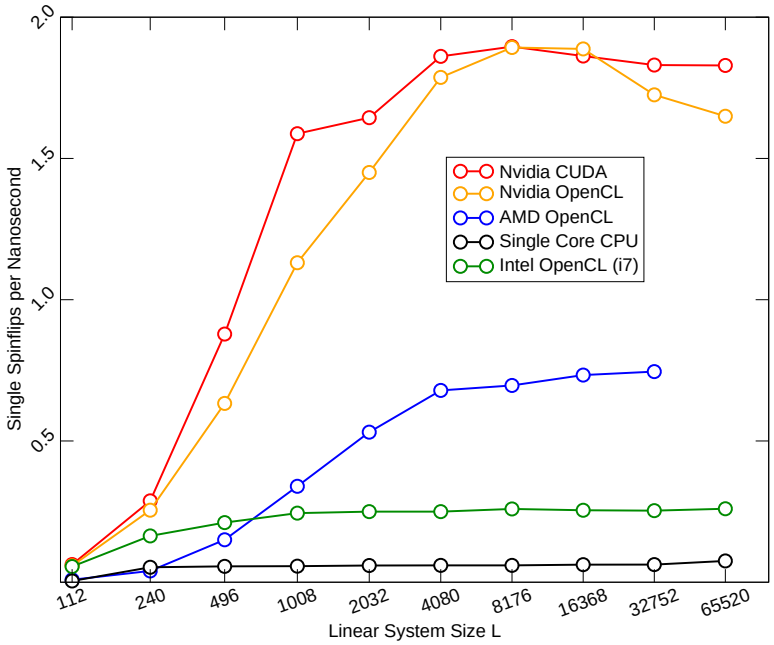


Figure 3.10: Single spin flips per nanosecond for a two dimensional system. Performance of OpenCL and CUDA on the Nvidia card are more or less comparable, while AMD clearly lacks behind. Also the largest system size fails to start for an unknown reason. This might be a due to an architectural limitation on AMD cards. Intels OpenCL implementation scales very well and impressively with the amount of cores on the i7 Nehalem processor. The figure was taken from [76].

The general idea of a canonical simulation is to replace the spin flip move with a spin exchange move. This means that for each spin that is flipped up, there has to be another spin flipped down in the lattice and vice versa.

3.6.1 Fisher-Yates Shuffle

The selection of spins needs to be done randomly, but in a way that we can make sure that no spin and one of its neighbors ever gets updated at the same time. One solution to this problem is to subdivide the spin lattice into blocks, and pair them up prior to execution of the update *kernel*. The problem of assigning each block exactly one partner in a way that each partner exactly gets chosen once is solved by the Fisher-Yates shuffle [112]:

```
1 for (int32 i = 0; i < BLOCKS; ++i)
2 {
3   shuffle[i] = i;
4 }
```

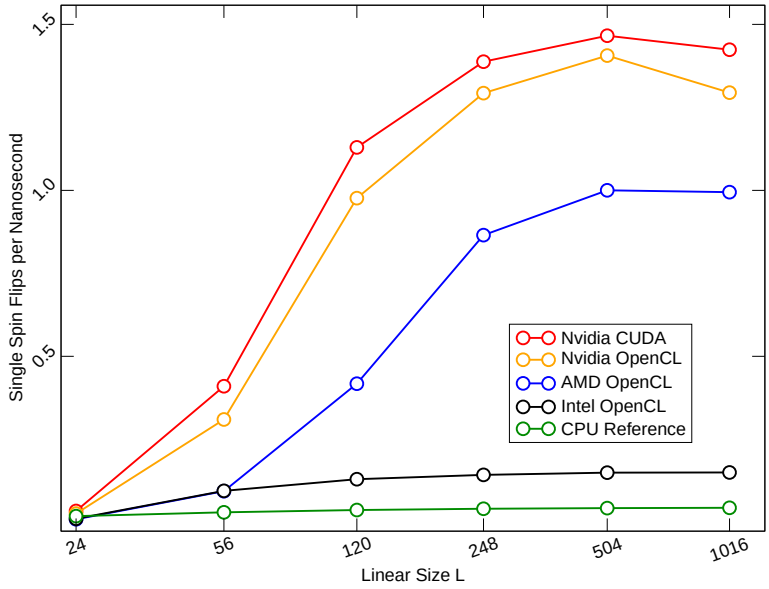


Figure 3.11: Single spin flips per nanosecond for a three dimensional system. The Nvidia card fares again nearly equally well on CUDA and OpenCL. AMDs performance is not much behind in this case, making the use of an AMD card a true alternative. The figure was taken from [76].

```
5
6   for (int32 i = BLOCKS - 1; i > 0; --i)
7   {
8     int j = rand(i+1);
9     int temp = shuffle[i];
10    shuffle[i] = shuffle[j];
11    shuffle[j] = temp;
12  }
```

We found that using a LCRNG does affect the quality of the simulation (in fact it introduced a systematic error in our case), so a better random number generator is strongly advised here. Also special care has to be taken because it is easy to introduce a *modulo bias* when selecting random number in a range of $[0; (i+1)]$. Generating a random number in a variable range guarantees that some of these ranges will not evenly divide the natural range of the random number generator. The remainders will not be evenly distributed but distributed in favor of smaller numbers.

3.6.2 Ergodicity

A problem that has to be taken care of here is to make sure that spin exchange is happening between all subsystems of the lattice. Even the checkerboard algorithm used for parallelization of the Ising model Monte Carlo update does strictly not fulfill ergodicity, since when exchanging spins, spins are only exchanged between black and black (white and white) sites of the lattice.

This can be solved in theory by mixing regular truly random exchange moves in between the parallel lattice updates once every N steps and choose N so that it does not affect GPU performance.

3.7 Hardware Abstraction

We aimed for a versatile and truly platform independent implementation of the Ising model that performs efficient in two as well as in three dimensions. This section has been published word for word in [73].

Platform independence is achieved by abstracting away API specific implementation details and reusing the update *kernel* for different APIs. Performance is comparable between OpenCL and CUDA on the same hardware (Nvidia), however there are major differences between implementations of different vendors.

Performance could in theory be improved by making explicit use of the vector hardware that underlies the AMD architecture. This would however sacrifice portability and would need another round of optimization.

This section highlights the interfaces used to abstract from the different hardware platforms used.

Device (IDevice)

The device implements the basic functionality of setting up the parallel computing device in CUDA or OpenCL. The most important functionality the IDevice interface needs to provide is LoadSource(). In OpenCL, it reads a OpenCL program source file, compiles it, parses it for *kernels* and exposes the *kernels* by their names as Device Programs (see below).

In CUDA, runtime compilation is not supported, so we are limited to reading precompiled compiled device modules (using cuModuleLoad).

Device Program (IDeviceProgram)

Programs that are executed on the device come in very different flavors. In early APIs for programming graphics processing devices, it was only possible to program them in a machine language or very specific custom code. Today, most of the GPGPU/massively parallel programming environments provide a way to

specify them in a C-like language. This allows for writing a preprocessor (or using the C preprocessor) to wrap the implementation specific differences into macros that hide away the platform dependent differences.

A major difference between CUDA and OpenCL is that CUDA code cannot be compiled at runtime, but has to be compiled to byte code before the program is executed. Each program can be passed input parameters, that are then mapped to function arguments of the *kernel*. These can be either POD types or pointers into device memory. The functions SetInputParameter and SetInputContainer are used for this purpose. The basic interface for the device program can be expressed very simply:

SetInputParameter, this is called before the *kernel* is run, and can be used to pass small data (e.g. the temperature T of the system) to the *kernel*.

SetInputContainer, this is also called before *kernel* execution and is used to give the *kernel* access to large areas of memory (e.g. the spin field in our Ising simulation). This large area of memory is managed by a Data Container. (see section 3.7).

Run, after all Parameters and Containers are passed to the *kernel*, it is executed with a single call to Run.

This interface can be is implemented by a CCUDADeviceProgram and an CCLDeviceProgram which internally deal with the API specific differences. For CUDA, we use the Device API, so we are free to load different *kernels* at runtime. The *kernels* have to be compiled to bytecode as opposed to OpenCL, where the *kernels* are compiled directly from source.

Data Container (IDataContainer)

CUDA as well as OpenCL provide similar ways to allocate storage space on the device memory and transfer data between memory spaces in the host memory. The data container is an abstraction of storage space on the device, with a mapped memory on the host device that can be used in a number of different Data Access Modes. The data container is created with a given size and dimensionality as a parameter. Upon construction, the Data Container allocates the needed memory space on the host, as well as on the device.

We define several access modes for allocated memory on the GPU:

DA_READONLY, which is for memory that can only be written to from the host and only read by the device

DA_WRITEONLY, which is for memory that can only be written to by the device and is not accessible for reads from the device. It is used for pure output data from the *kernel* execution.

DA_READWRITE, this is two way memory that can be written and read directly from the device during *kernel* execution.

There are two important functions which the IDataContainer interface has to provide.

HostLock(), this locks memory on the host and returns a pointer to the memory. This means when this function is called, the container will make sure that if the memory has been locked on the device since the last call to HostLock(), it will copy over the latest data from the device down to the host.

DeviceLock(), if a *kernel* is called with a Data Container as input, a call to DeviceLock() will be done. If the last call was to HostLock(), or no call at all, the memory will be copied to the device, so it is accessible from the *kernel*. If DeviceLock(has been called before, no copying is done. If HostLock() is called after DeviceLock() the memory will be copied back to the host.

This behavior guarantees that whatever owner (host or device) locked the memory last, will obtain the latest version of the data in the container.

3.7.1 Kernel Definition

The behavior of each Device Program is declared in a *kernel* definition. As stated before, they can be declared in a C-style language. We are using the built-in preprocessor of each implementation of the C-like language to translate the *kernel* definition into the native language (CUDA/OpenCL). To hide away platform dependent differences, each *kernel* definition file has to include "unified-kernelint.h" because here, all the necessary preprocessor macros are defined.

Header

A function is defined by the following platform independent sequence which uses the macros **DEFINE_KERNEL** and **END_DEFINE_KERNEL**:

```
DEFINE_KERNEL(name) argument list END_DEFINE_KERNEL {body}
```

This sequence translates into one of the following, dependent on the API in use:

CUDA
```
extern C __global__ void name(argument list) { body }
```

OpenCL
```
__kernel void name(argument list) { body }
```

Access to Execution Coordinates from Within the Kernel

The local and global coordinates of a *kernel* that is executed in a grid of threads are accessible via the macros

`GROUP_ID_X, GROUP_ID_Y, GROUP_ID_Z`

For the global coordinates and

`LOCAL_ID_X, LOCAL_ID_Y, LOCAL_ID_Z`

for the local coordinates.

On CUDA, this maps to blockIdx.[x,y,z] and threadIdx.[x,y,z], while on OpenCL it maps to get_group_id(0-2) and get_local_id(0-2).

Chapter 4

Interfaces in the 3D Ising Model

4.1 Introduction

While the previous chapter has demonstrated that an efficient parallel GPU implementation can outperform a non-parallel CPU version by orders of magnitude, this chapter sets on to prove that these numbers have significance beyond synthetic benchmarks. We will use it to improve on previous investigations in the 3D Ising model, especially those by Winter et. al. [41] and Kim et. al. [42].

It turns out that the additional resources we have through the implementation detailed in chapter 3 combined with some new methodical ideas described in chapter 2, enabled us to come up with a range of new results in understanding the interfacial properties of a two-phase Ising system with an interface.

We take a close look at the mechanisms that induce the roughening transition (section 1.3) which occurs at a finite temperature T_R in the 3D Ising model. This transition is believed to be induced by an appearance of a *Step Free Energy* at low temperatures, where the thermal fluctuations are large enough to overcome the underlying structure of the lattice and interfaces align along a lattice plane. The scaling behavior of the *Step Free Energy* which has previously been investigated in SOS models is reproduced in our system in section 4.7. The results are able to reproduce the theoretical predictions of Fisher et. al. [25] concerning the surface stiffness to great accuracy in section 4.6 which has not been done to this extend in numerical simulations. A line tension contribution is investigated in section 4.9 and a contact angle dependence of the line tension could be verified.

To understand the free energies related to the different surfaces that are present in the system under investigation, we have a closer look at the geometry, and introduce some necessary variables.

4.2 Interfacial Free Energies

In figure 4.1, the geometry of the system is specified. This special geometry allows us to investigate a surface between a majority-up phase $(+)$ and a majority-down $(-)$ phase which is inclined by an angle Θ with respect to a crys-

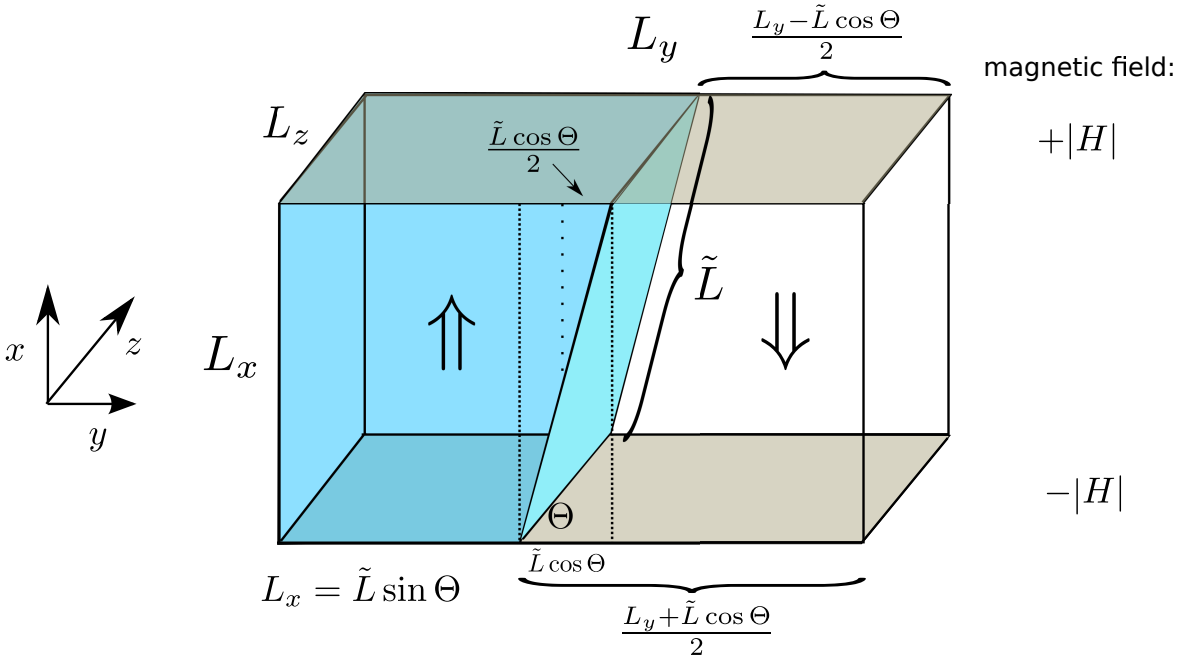


Figure 4.1: The simulation geometry. Over the boundaries of the y-dimension, (generalized) anti-periodic boundary conditions are transmitted. The boundary conditions in y-direction stabilize a flat interface between the two phases. In x-direction, the simulation domain is bounded by hard walls, which can be adjusted to a external wall field of magnitude $|H|$, so the uppermost and lowermost layers of spins feel the influence of the wall field. When an external H-field is applied to the Ising-spins next to the wall, the interface is tilted with an angle Θ with respect to the wall surface. If the wall field is adjusted so the upper field has a value of $+H$, while the lowermost has a value of $-H$, the symmetry does not favor any of the two phases and the interface gets tilted without a systematic drift in any direction.

tal axis. In z-direction, there are periodic boundary conditions, so the system is translationally invariant in this direction. Our intention is to stabilize a single surface, which is achieved with (generalized) antiperiodic boundary conditions (see sections 2.2.3 and 2.2.4).

In the y-z plane, we have applied a magnetic field of magnitude $|H|$. One wall has a magnetic field of $+H$, while the other one has a magnetic field of $-H$, so no drift is introduced. This symmetry needs to be reflected by the boundary conditions in y-direction. The antisymmetric wall field tilts the surface by an angle $(90° - \Theta)$. This breaks the translational symmetry in x-direction, so that we have to generalize the periodic boundary conditions in y-direction (see section 2.2.4).

The magnitude of this field $|H|$ has to be chosen such that one stays in the

partial wetting regime (section 1.4.1) of the Ising model. If one would choose $|H|$ so large that one would be in the regime of complete wetting (section 1.4.1), the interface would be parallel to the walls (for $L_x \to \infty$) i.e. ($\Theta = 0$).

We use $\gamma(\Theta)$ for the interface tension of a planar interface between coexisting phases in dependence of the contact angle Θ. Note that the interface is only tilted with respect to the y-z plane. The most general case where these contact lines are tilted relative to the x-axis by another angle ϕ, is not studied for simplicity ([113] however studies the interface tension anisotropy in the 111 interface in the 3D ising model).

This interface tension $\gamma(\Theta)$ between the $(+)$ and the $(-)$ phases depends on the angle Θ relative to lattice direction (angle with interface normal is $90^\circ - \Theta$) wall tensions $\gamma^{(\pm)}(\pm|H|)$, where the upper index refers to the sign of the spontaneous magnetization. A suspected line tension contribution is denoted by $\tau(|H|, \Theta)$.

This leads to a total interfacial free energy contribution which takes into account all interfaces in the system

$$\begin{aligned} F_{\text{int}} &= \tilde{L} L_z \gamma(\Theta) \\ &+ L_z \left(\frac{L_y + \tilde{L}\cos\Theta}{2} \right) \gamma^{(+)}(+|H|) \\ &+ L_z \left(\frac{L_y - \tilde{L}\cos\Theta}{2} \right) \gamma^{(-)}(+|H|) \\ &+ L_z \left(\frac{L_y - \tilde{L}\cos\Theta}{2} \right) \gamma^{(+)}(-|H|) \\ &+ L_z \left(\frac{L_y + \tilde{L}\cos\Theta}{2} \right) \gamma^{(-)}(-|H|) \end{aligned} \tag{4.1}$$

When we change the signs of the magnetization and of the fields, the Ising system stays invariant. This leads to the following symmetries:

$$\begin{aligned} \gamma^{(+)}(+|H|) &= \gamma^{(-)}(-|H|) \\ \gamma^{(-)}(+|H|) &= \gamma^{(+)}(-|H|) \end{aligned} \tag{4.2}$$

These symmetries allow simplification of equation 4.1 which leads to

$$\begin{aligned} F_{\text{int}} &= \tilde{L} L_z \gamma(\Theta) \\ &+ L_z \left(L_y + \tilde{L} \cos\Theta \right) \gamma^{(+)}(+|H|) \\ &+ L_z \left(L_y - \tilde{L} \cos\Theta \right) \gamma^{(+)}(-|H|) \end{aligned} \tag{4.3}$$

using $\tilde{L} = L_x / \sin\Theta$. Here, the constraint that the total magnetization is zero has been used by the symmetry already.

$$\begin{aligned} F_{\text{int}} &= L_x L_z \gamma(\Theta) / \sin\Theta \\ &+ L_z (L_y + L_x / \tan\Theta) \gamma^{(+)}(+|H|) \\ &+ L_z (L_y - L_x / \tan\Theta) \gamma^{(+)}(-|H|) \end{aligned} \tag{4.4}$$

The free energy needs to be determined first. When determined, it can be used to extract the different contributions and dependencies from it to gain more knowledge about the nature of the interface.

4.3 Free Energy Integration

There are plenty of different methods to extract free energies from a physical system. We will use a method that is called *Thermodynamic Integration*. The idea is to start at a point in phase space for which the free energy has a known value, and integrate over a certain path in phase space to the point we are interested in. In our case, this path starts with a $\Theta = 90^\circ$ configuration from a zero temperature state. Then it moves along the temperature axis towards a specific temperature to obtain the values of the surface tension and the line tension for an interface aligned along the grid. From there, the integration is done along the path where the temperature is kept constant, and the wall field H is increased in even steps to obtain the free energy of a configuration with a certain angle that matches the H field. The path in configuration space is depicted in figure 4.2.

4.3.1 With Respect to Temperature

In this section we show how thermodynamic integration with respect to temperature can be used to extract the free energy of the system

$$F(T) = U(T) - TS \tag{4.5}$$

therefore

$$\beta F(\beta) = \beta U(\beta) - S/k_B$$

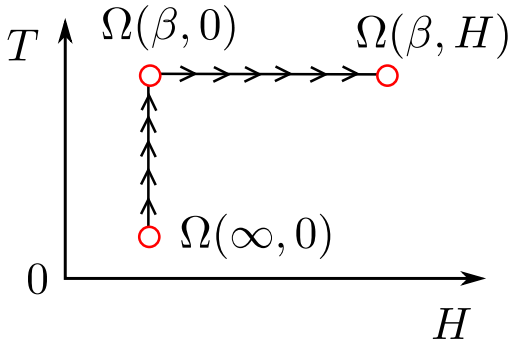


Figure 4.2: The integration path in the T-H plane for the thermodynamic integration. First, an integration along the temperature axis is performed, after that comes the integration along the external field H.

with $\beta = 1/k_B T$.

Differentiation leads to

$$\frac{\partial(\beta F)}{\partial \beta} = U(\beta)$$

which is an observable quantity in our system. Integrating again over β leads to the following expression for a free energy difference at two temperatures β_1 and β_2.

$$\beta_2 F(\beta_2) - \beta_1 F(\beta_1) = \int_{\beta_2}^{\beta_1} U(\beta') \mathrm{d}\beta'$$

The last expression leads us to the observation that to get an explicit value for $F(\beta)$, we do not only need the full temperature dependence of $U(\beta)$, but also a known value for the free energy $F(\beta)$ at a known temperature β_2.

A known value for $F(\beta)$ is

$$F(\beta \to \infty) \to U(\beta \to \infty).$$

In this case, the free energy is the same as the Internal Energy of the system. Since $U(\beta)$ differs at large β from $U(\beta \to \infty)$ by terms of order $\exp(-6\beta J)$ in the bulk and $\exp(-4\beta J)$ at a planar interface, it suffices to choose β_2 finite but large enough so that these terms ($\exp(-6\beta J)$, $\exp(-4\beta J)$) are neglibibly small. We now consider the difference in free energy between two systems which both have linear dimensions $L_x \times L_y \times L_z$. One system has periodic boundary conditions in y and z directions, and hard walls in y-z plane and a free energy F_p. The other system has antiperiodic boundary conditions in y-direction and a free energy F_a. We now consider the difference

$$F_p - F_a = \Delta F = L_z L_x \gamma_{vl} + 2 L_z \tau.$$

We can also exploit the fact that the interfacial free energy (and the line tension) are strictly zero in the regime of the disordered phase. The bulk free energies of the phases with positive and negative magnetization in the bulk are the same, and therefore contributions from the bulk free energies cancel when one considers the difference $F_p - F_a$. For $H = 0$ the wall free energies of the two systems cancel out as well. So only contributions due to the surface tension γ_{vl} and the line tension τ remain. If we chose a system with $\beta_2 < \beta_c$ above the critical temperature, the free energies of the two systems (periodic and partly antiperiodic) are the same, so the internal energy difference reduces to a difference in free energies at the end temperature β_1.

$$\int_{\beta_2}^{\beta_1} \left[U_p(\beta') - U_a(\beta')\right] \mathrm{d}\beta' = \beta_1(F_p(\beta_1) - F_a(\beta_1)) - \beta_2(F_p(\beta_2) - F_a(\beta_2))$$

$$= \beta_1(F_p(\beta_1) - F_a(\beta_1))$$

Of course, β_2 must be chosen much smaller than β_c, so finize size effects of order $\exp(-L_y/\xi)$ where ξ is the bulk correlation length are negligibly small.

Let us consider two different systems with sizes

$$L_x^{(1)} \times L_y \times L_z$$

and

$$L_x^{(2)} \times L_y \times L_z$$

Using the free energy difference ΔF we can extract the absolute value of the surface tension, if we vary L_x and keep L_z and L_y constant:

$$\Delta F^{(2)} - \Delta F^{(1)} = L_z \gamma_{vl} \left(L_x^{(2)} - L_x^{(1)}\right),$$

which makes it possible to extract the surface tension γ_{vl}

$$\gamma_{vl} = \frac{1}{L_z} \cdot \frac{\Delta F^{(2)} - \Delta F^{(1)}}{L_x^{(2)} - L_x^{(1)}} \tag{4.6}$$

The surface tension γ_{vl} is shown in figure 4.3 for different temperatures.

We can also extract the line tension τ, by varying L_z and keeping L_y and L_x constant

$$\Delta F^{(2)} - \Delta F^{(1)} = \left(L_z^{(2)} - L_z^{(1)}\right)(L_x \gamma_{vl} + 2\tau)$$

if we know γ_{vl}:

$$\tau = \frac{1}{2}\left(\frac{\Delta F^{(2)} - \Delta F^{(1)}}{L_z^{(2)} - L_z^{(1)}} - L_x \gamma_{vl}\right) \tag{4.7}$$

This integration has been carried out by Kim et. al. [42], the results have been verified and are used as a starting point to calculate the free energy contributions of the interfaces when the translational symmetry in x-direction is broken by an external wall field as elaborated on in the next section. The results for τ are shown in figure 4.3.

4.3.2 With Respect to the Wall Field

The free energy of a system with generalized antiperiodic boundary conditions in y-direction and a wall field $H \neq 0$ and the free energy of the same system with $H = 0$ are compared. (Recall that for $H = 0$ APBC and GAPBC are equivalent.) A suitable variation in system sizes in all three dimensions leads to expressions that allow to extract the difference in surface tensions $\gamma(H \neq 0) - \gamma(H = 0)$ (section 4.5) and in the line tensions $\tau(H \neq 0) - \tau(H = 0)$ (section 4.9).

$$F_s(H) = F_s(H = 0) - HM_s \tag{4.8}$$

Here, M_s is the magnetization in the layer on which the surface field H acts. Then in analogy with the well-known relations for the bulk we have $M_s = -(\partial F(H)/\partial H)_T$. In the case of wall fields $H \neq 0$, the difference in Surface Free Energy to the system with no wall field can be estimated using thermodynamic integration

$$\Delta F_s(H) = F_s(H) - F_s(0) = \int_0^H \mathrm{d}H' \langle \frac{\partial F_s(H')}{\partial H'} \rangle = -\int_0^H \mathrm{d}H' \langle M_s(H') \rangle, \tag{4.9}$$

where $M_s(H)$ denotes the surface magnetization of both surfaces added together. Following from the symmetry of the system, the total average surface magnetization should be two times the magnetization of one surface. It follows the simple definition

$$M_s = \#\uparrow - \#\downarrow$$

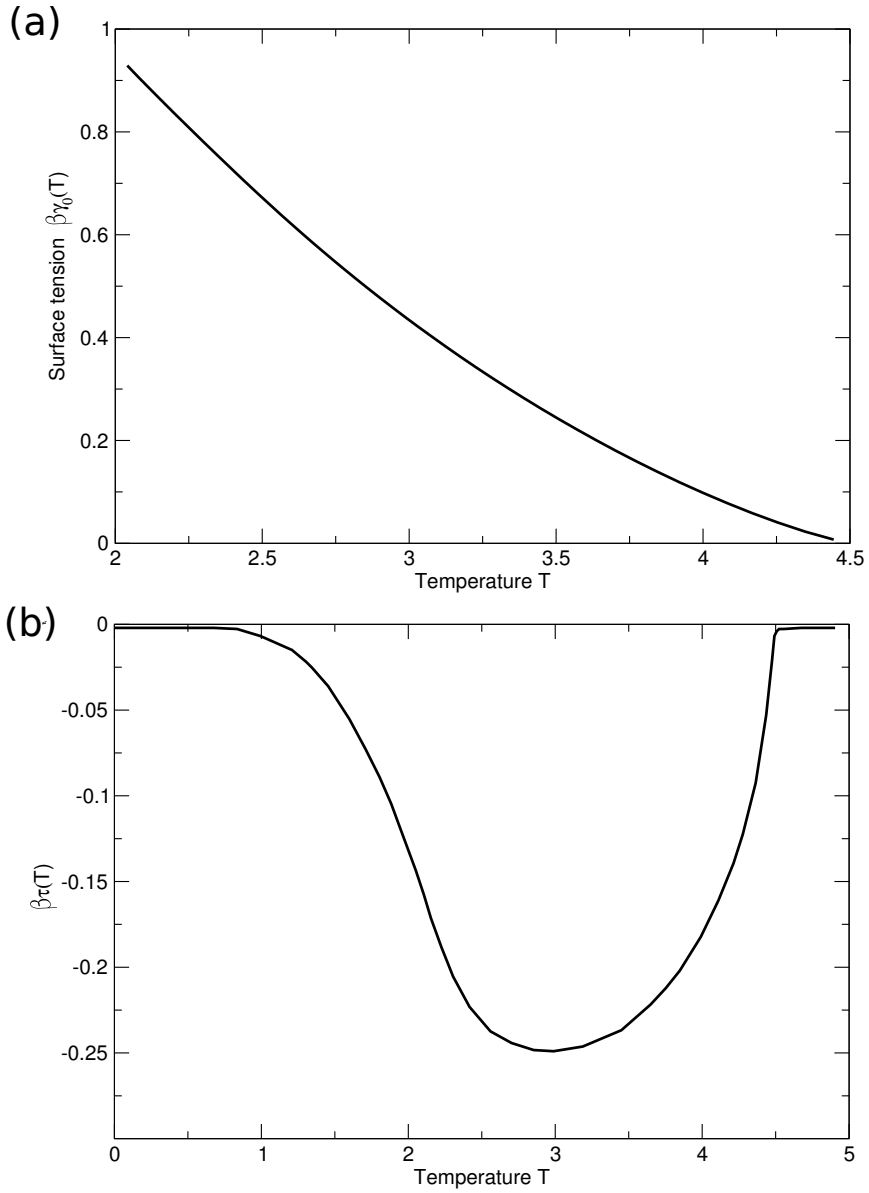


Figure 4.3: The surface tension $\gamma_0(T)$ (a) for different temperatures from the equation 4.6 as calculated by Hasenbusch et. al. [114]. This data has been verified in our simulation and used as a basis for the integration of the angle dependent tension in section 4.5 and the stiffness calculation in section 4.6. The line tension $\tau(T)$ (b) for different temperatures from the equation 4.7 as calculated by Kim et. al. [42]. The line tension data is used in section 4.9 to construct the contact angle dependent line tension $\tau(\Theta, T)$.

4.4 Contact Angle

If an external H-field is applied to the Ising spins next to the wall, the contact angle between the surface and the walls is expected to change from 90°, so the interface is no longer aligned with the lattice orientation.

The H-field dependence of this contact angle has been investigated by Winter et. al. [27] using Young's Equation [26] in its standard form which ignores any dependence of $\gamma_{vl}(T)$ on the orientation of the interface relative to the lattice axes

$$\gamma_{vl}(T)\cos(\Theta) = f_s^{(+)}(T,H) - f_s^{(-)}(T,-H), \tag{4.10}$$

where $f_s^{(+)}(T,H)$ and $f_s^{(-)}(T,-H)$ are the surface free energies attributed to the part of both phases which are in contact with the upper and the lower wall. We first provide a derivation of Young's Equation from our geometry.

4.4.1 Derivation of Young's Equation

Young's equation should be a result of minimizing the free energy of the system with respect to the contact angle Θ. The angle Θ is now given by the condition that $\partial F_{\text{int}}/\partial\Theta = 0$ or in other words, that $F_{\text{int}}(\Theta)$ must be at a minimum.

To this end, we use the following approximation:

$$\gamma(\Theta) = \gamma \quad \text{independent of} \quad \Theta,$$

and therefore

$$\frac{\partial\gamma}{\partial\Theta} = 0 \quad \text{for all angles} \quad \Theta.$$

It is not clear that this approximation is justified, and in fact we will see later that it is wrong in our case. It becomes correct only close to the critical point of the bulk, where the correlation length of the magnetization fluctuations becomes much larger than the lattice spacing and all corrections to the critical scaling behavior due to lattice anisotropy become negligible. However, we want to study the region well below the critical temperature. For the purpose of this section, we assume the approximation to be true, so the minimization of the free energy $F(\Theta)$ (equation 4.4) yields

$$\begin{aligned}\frac{\partial F_{\text{int}}}{\partial\Theta} = 0 = \quad & -L_x L_z \gamma \frac{\cos\Theta}{\sin^2\Theta} \\ & -L_x L_z \gamma^{(+)}(+|H|)\frac{1}{\sin^2\Theta} \\ & +L_x L_z \gamma^{(+)}(-|H|)\frac{1}{\sin^2\Theta}\end{aligned} \tag{4.11}$$

which leads to

$$\Rightarrow \gamma \cos\Theta = \left[\gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|)\right] \tag{4.12}$$

which is the familiar form of the Young's equation [26]. We are not able to measure the quantity

$$\left[\gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|)\right]$$

in equation 4.12 directly in our present geometry. So, to use Young's equation for an estimate of the contact angle in our system, we use a different system geometry. Winter et. al. [41] describe a system with periodic boundary conditions which allows easy extraction of the mean magnetizations of the uppermost ($\langle m_o^{(-)} \rangle$) and lowermost ($\langle m_u^{(-)} \rangle$) layers of spins. These are needed for calculation of the free energies in equation 4.10. We follow along the same line, but with much larger systems. We use our grand canonical GPU implementation to simulate a system with periodic boundary conditions (figure 4.4) and with the size

$$88 \times 504 \times 504$$

which allows for investigation of a large range of angles and two larger sizes

$$184 \times 504 \times 504 \quad \text{as well as} \quad 504 \times 504 \times 504$$

for closer investigation of the behavior at small angles, since for small angles Θ finite-size effects are more pronounced. Simulations are done at different temperatures ranging from well below the roughening transition temperature T_R ($T = 1.5$) up to temperatures near the bulk critical temperature T_C ($T = 4.2$). The magnetization difference $m_o - m_u$ is averaged over a large number of Monte Carlo steps (100.000). A measurement for a system of size $88 \times 504 \times 504$ can be seen in figure 4.5. *Thermodynamic Integration* (section 4.3) yields the data in figure 4.6. This is put in equation 4.12 to extract the surface angle Θ. The results of this procedure are shown in figure 4.7.

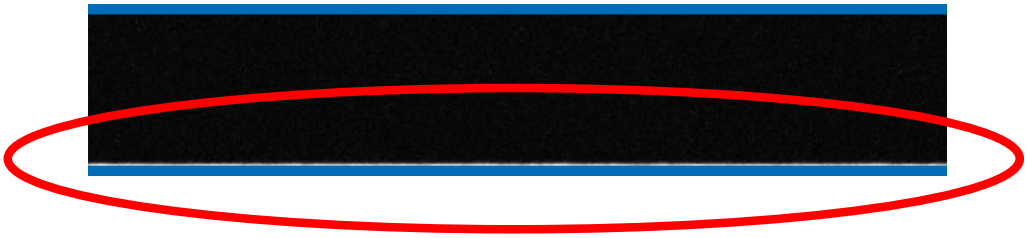


Figure 4.4: A *grandcanonical* simulation of a system ($88 \times 504 \times 504$) with periodic boundary conditions (PBC) does not form an interface. The wall field of $H = 0.8 \neq 0$ creates a visible thin film of up spins in the lower layer of spins (red circle). The magnetization of the upper layer of spins $\langle m_o^{(-)} \rangle$ is then subtracted from the magnetization of the lower layer $\langle m_u^{(-)} \rangle$ to extract the contact angle using Young's equation.

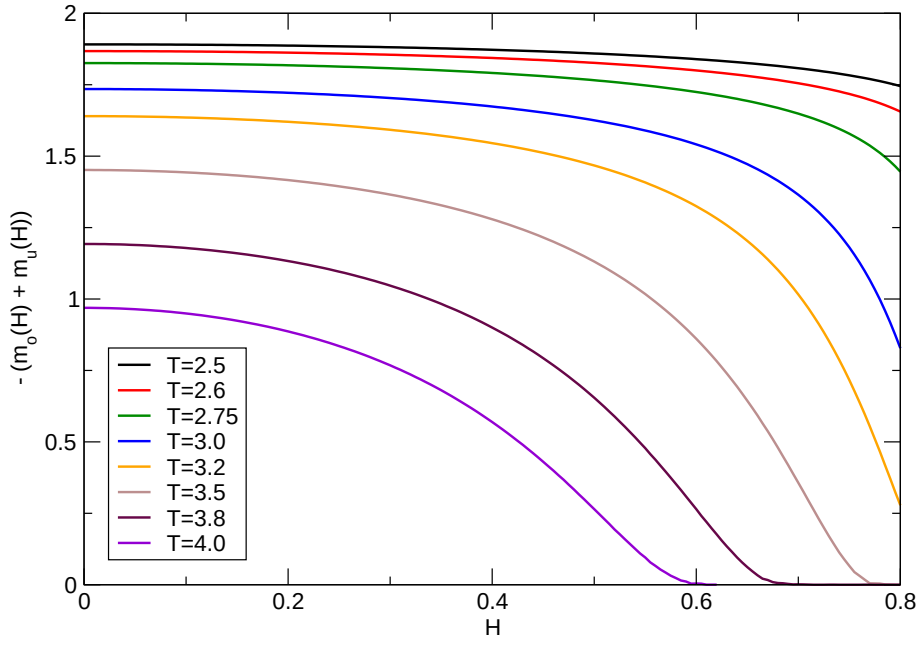


Figure 4.5: The magnetization difference $\langle m_o^{(-)} \rangle - \langle m_u^{(-)} \rangle$ of the uppermost and the lowermost layer. Thus when integrated over, this difference can be used to extract the free energy difference seen in figure 4.6

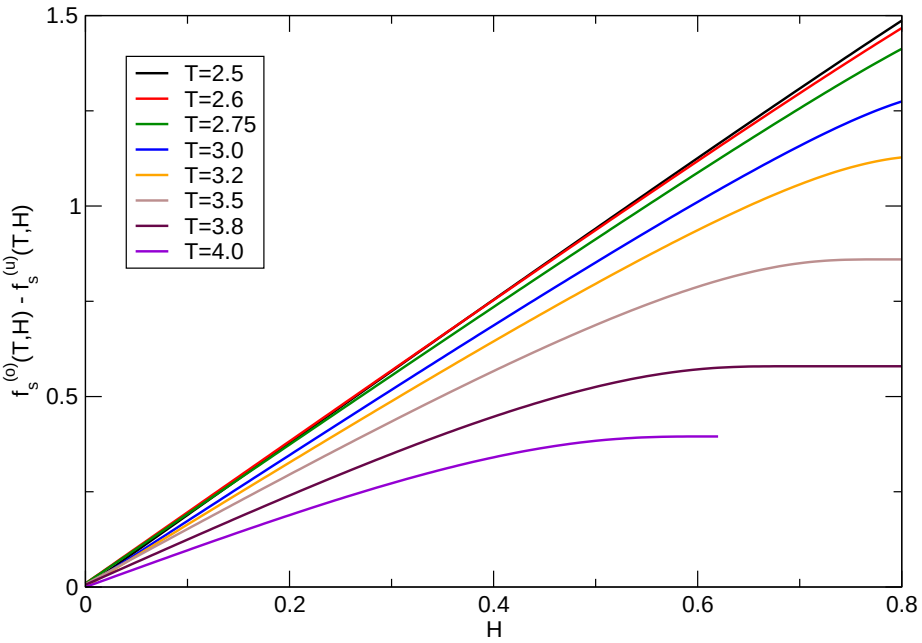


Figure 4.6: Subtracting the integral of the magnetization of the upper layer from the integral of the magnetization of the lower layer leads to the total free energy of the upper layer and the lower layer combined. This free energy is the free energy attributed to the interfaces of the spins at contact with the wall.

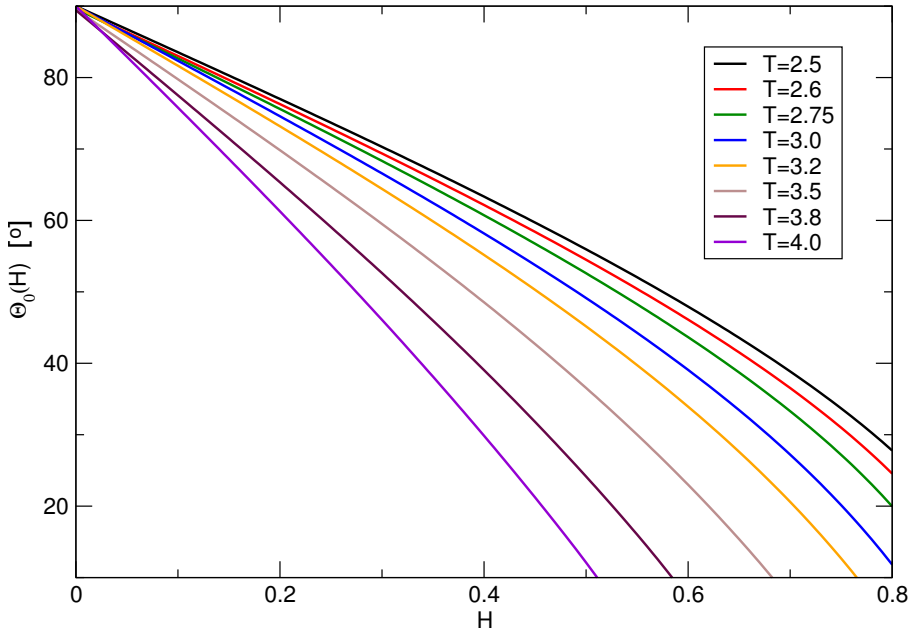


Figure 4.7: Using equation 4.10, the contact angle can be extracted from the total surface free energy of the system, under the assumption of an isotropic surface tension. The measured data is in full agreement with Winter et. al. [27] for the temperatures that are available for comparison. Simulations are extended to various temperatures down to a temperature of $T = 1.5$

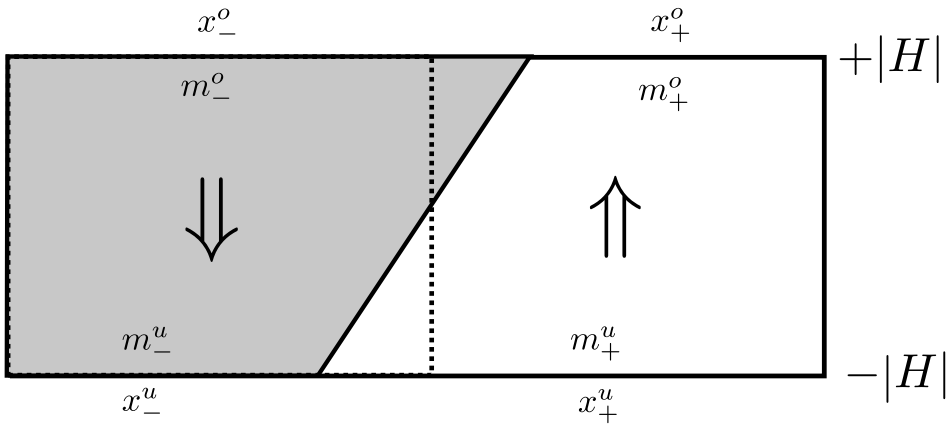


Figure 4.8: The geometry and definitions of a two domain state system with antiperiodic boundary conditions in y-direction and a tilted interface between them. The lengths of the spin domains in y-dimension in the uppermost and lowermost layer are denoted by x^o_- and x^u_- for the spin-down majority phase, and x^o_+ and x^u_+ in the spin-up majority phase.

4.4.2 Without Young's Equation

As we came to realize, the original form of the Young's equation does not hold in our case, because the assumption

$$\gamma'(\Theta) = 0$$

is not justified. We need a modification for it to hold under these circumstances. We will now detail a different method to estimate the contact angle by only using geometric arguments and no further assumptions on the model. The system is expected to form a two phase configuration like detailed in figure 4.8.

For this system, a *domain state magnetization* can be extracted from the uppermost ($m_o^{(\text{d.s.})}$) and the lowermost ($m_u^{(\text{d.s.})}$) layer (the layers next to the walls), which is defined as

$$m_o^{(\text{d.s.})} = (x^o_- / L_y) \cdot m^o_- + (x^o_t / L_y) m^o_+ \tag{4.13}$$

$$m_u^{(\text{d.s.})} = (x^u_- / L_y) \cdot m^u_- + (x^u_t / L_y) m^u_+. \tag{4.14}$$

It needs to be verified, under which boundary conditions the assumptions $m^o_- = const$ and $m^u_- = const$ are justified. The implications of the boundary conditions used on the surface magnetizations is described in sections 2.2.3 and

2.2.4. In short, antiperiodic boundary conditions can be used if caution is applied, the best boundary conditions to be used are our generalized antiperiodic boundary conditions, since they perfectly reflect the symmetry of the system.

Using equation 4.13 and the fact that $m_+^o = -m_-^u$ (see section 2.2.4) we can write

$$m_o^{(\text{d.s.})} = (x_-^o / L_y) m_-^o + (x_+^o / L_y) m_+^o = (x_-^o / L_y) m_-^o - (x_+^o / L_y) m_-^u$$

and using $x_+^o = L_y - x_-^0$

$$= (x_-^o / L_y) m_-^o - \frac{L_y - x_-^o}{L_y} m_-^u$$

$$= -m_-^u + \frac{x_-^o}{L_y}(m_-^o + m_-^u).$$

$$\rightarrow x_-^o = L_y \frac{(m_0^{(\text{d.s.})} + m_-^u)}{(m_-^o + m_+^u)}. \tag{4.15}$$

The same argument reveals for the lower side:

$$x_-^u = L_y \cdot \frac{(m_u^{(\text{d.s.})} + m_-^o)}{(m_-^u + m_-^o)} \tag{4.16}$$

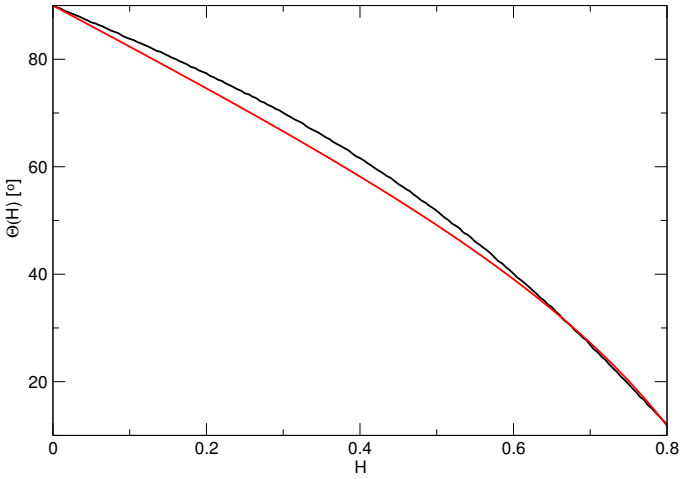
To obtain the contact angle now, the relation $x_-^o - x_-^u = D \cdot \tan(\Theta)$ can be used:

$$\Theta = \arctan\left(\frac{x_-^o - x_-^u}{D}\right) \tag{4.17}$$

Note that we do not imply in this analysis that the interface between the coexisting phases can be microscopically described as a plane, and we do not imply that along the walls the local magnetizations are constant $(m_-^o, m_+^o, m_-^u, m_+^u)$ up to the contact line. Our analysis *defines* the interface as a dividing plane between bulk phases, and *defines* the contact line as the intersection of this plane with the wall. Thus our methods do not imply anything about the structure of the contact line, of course.

Our simulations show that the contact angle calculated via this construction are systematically larger than the predictions of Young's equation, a result that can be seen for one temperature in figure 4.9. A complete assembly of results above the roughening transition temperature is shown in figure 4.10.

Figure 4.9: The contact angle for $T = 3.0$ at a system size of $84 \times 504 \times 504$, calculated by the geometric construction in equation 4.17 (in black) is systematically larger than the angle calculated by means of Young's equation (in red). This is a first hint that Young's equation does clearly not hold under these circumstances. We will see that this difference is very valuable to obtain the angle dependence of the surface tension $\gamma(\Theta)$ as will be shown in section 4.5.



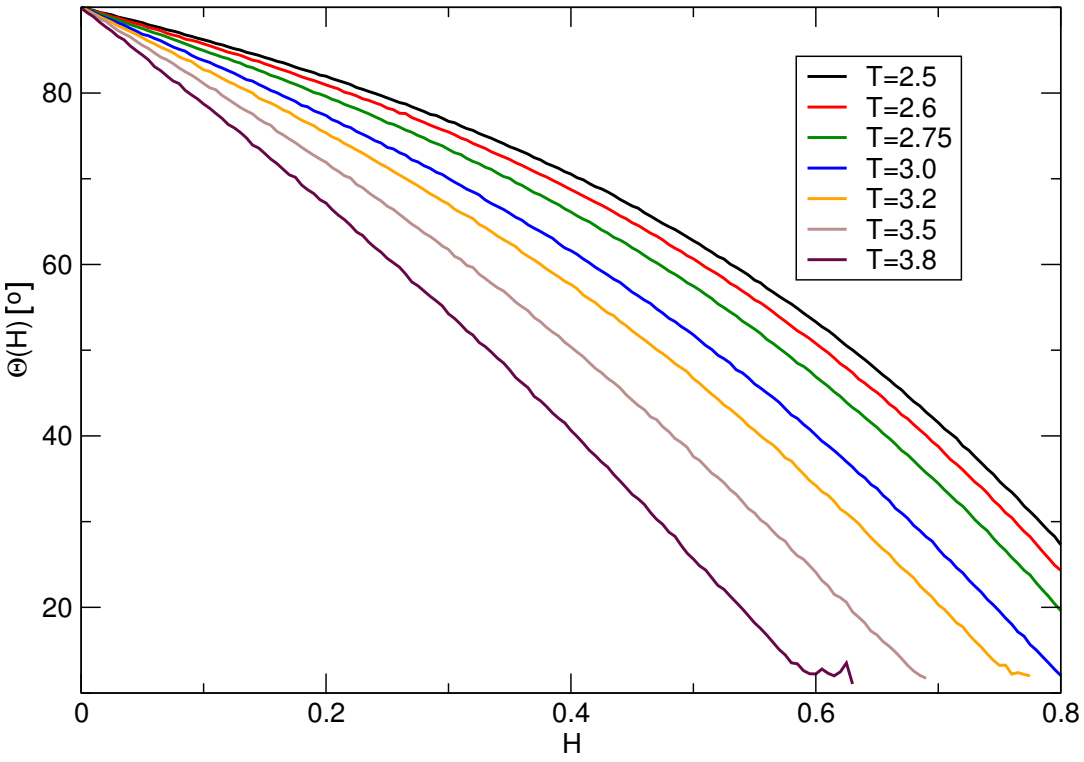


Figure 4.10: The contact angle calculated by equation 4.17 for dhifferent temperatures ranging from $T = 2.5$ up to $T = 3.8$. As revealed by figure 4.9, the curves are systematically larger than prediced by Young's equation (equation 4.10).

4.5 Surface Tension Anisotropy

At high enough temperatures, thermal fluctuations can overcome the underlying lattice structure as seen in figure 1.3, but at low temperatures the anisotropy of the lattice leads to an anisotropy of the surface tension. We will now discuss the effect of an angle dependent surface tension $\gamma(\Theta)$.

As discussed above in section 4.4.1, the contact angle that will be realized in equilibrium is calculated by minimizing the surface free energy $F_{\text{int}}(\Theta)$ with respect to Θ:

$$\left(\frac{\partial F_{\text{int}}}{\partial \Theta}\right)_{H,T} = 0$$

This time we do not assume that $\gamma'(\Theta) = 0$, so when deriving equation 4.4 a new $\gamma'(\Theta)$-term gets introduced.

$$\begin{aligned} 0 &= \frac{1}{\sin^2\Theta} \cdot L_x L_z \left[\left(\gamma'(\Theta)\sin\Theta - \gamma(\Theta)\cos\Theta \right) \right. \\ &+ \left. \left(\gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|) \right) \right] \end{aligned}$$

Therefore, we arrive at a modified Young's Equation which includes a term that is dependent on the derivative of the surface tension

$$\gamma(\Theta)\cos\Theta - \gamma'(\Theta)\sin\Theta = -\frac{\mathrm{d}}{\mathrm{d}\Theta}\left(\gamma(\Theta)\sin\Theta\right) = \gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|) \tag{4.18}$$

Note that the term on the right is exactly the term that was extracted from simulations in an effort to get the contact angle using the Young's Equation (section 4.4). We now use the abbreviation

$$\Delta\gamma = \gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|).$$

In case of an angle dependent line tension (see section 1.4.2 for more information) there can be an additional term involving $(\partial\tau/\partial\Theta)_H$, which can be of the same order as the term with γ', but in the limit $L_x \to \infty$ it will eventually disappear. First we solve equation 4.18 for the derivative $\gamma'(H)$

$$\sin\Theta\gamma'(\Theta) + \Delta\gamma = -\gamma(\Theta)\cos\Theta$$

$$\sin\Theta\gamma'(\Theta) = -\gamma(\Theta)\cos\Theta - \Delta\gamma$$

to arrive at a first order differential equation

$$\Rightarrow \gamma'(\Theta) = -\gamma(\Theta)\frac{\cos\Theta}{\sin\Theta} - \frac{\Delta\gamma}{\sin\Theta} \tag{4.19}$$

which can be integrated e.g. using standard Euler-Cauchy integration

$$\gamma(H + \Delta H) = \gamma(H) + \gamma'(H, \gamma(H))\Delta H$$

Since the integration errors are a problem except for extremely small ΔH, we decided to use a slightly more accurate predictor-corrector method as described here: First the predictor $\tilde{\gamma}(H + \Delta H)$ is calculcated,

$$\tilde{\gamma}(H + \Delta H) = \gamma(H) + \gamma'(H, \gamma(H))\Delta H$$

which is then in turn used to derive a more accurate formula for $\gamma(H + \Delta H)$:

$$\gamma(H + \Delta H) = \gamma(H) + \frac{1}{2}\Delta H\left(\gamma'(H, \gamma(H)) + \gamma'(H + \Delta H, \tilde{\gamma}(H + \Delta H)\right)$$

Equation 4.18 is integrated to obtain the surface tension $\gamma(\Theta)$ given a start value for $\gamma(90^\circ)$ which is obtained from the accurate results given in [114]. The original data was depicted before, in figure 4.3.

Figure 4.12 shows the integrated surface tension $\gamma(\Theta)$ over a range of angles. At high temperatures, the anisotropy is very small and eventually becomes negligible, for temperatures approaching and below the roughening transition (section 1.3), the anisotropy becomes more and more important and is expected to have an effect on nucleation rates in the Ising model and therefore of crystals in a simple cubic lattice in general. Also it should be noted that the effects of surface tension anisotropy can in some geometries be of the same order as line tension effects (section 1.4.2). Unlike line tension effects however, the effect of surface tension anisotropy will not disappear in the limit $L_x \to \infty$. To better quantify the anisotropy, it is calculated for an angle of 45° (table 4.11) and plotted explicitly by showing the ratio of surface tensions $\gamma(\Theta)/\gamma(90^\circ)$ starting from an interface aligned with the crystal plane (90°) up to an angle of 45° for different temperatures (figure 4.13).

Bittner et. al [113] have measured the surface tension anisotropy between the 100, 110 and 111 interface of the simple cubic Ising model. The 110 interface corresponds to a contact angle of 45° in our picture. Therefore, a direct comparison with their results is possible. Figure 4.14 compares our results to the results of [113] by direct comparison of the anisotropy $\gamma(45^\circ)/\gamma(90^\circ)$, showing good agreement between both approaches. We have seen that, similar to ice crystals,

Temperature T	Anisotropy $\gamma(45°)/\gamma(90°)$
2.5	1.050
2.6	1.042
2.75	1.032
3.0	1.021
3.2	1.014
3.5	1.008
3.8	1.004

Figure 4.11: Anisotropy values for different temperatures. This data is visualized and compared to literature in figures 4.14 and 4.15.

the Ising model features a surface tension anisotropy and we were able to quantify it over a large temperature range between the roughening temperature T_C and the bulk critical temperature T_C with higher accuracy than previous investigations (figure 4.15).

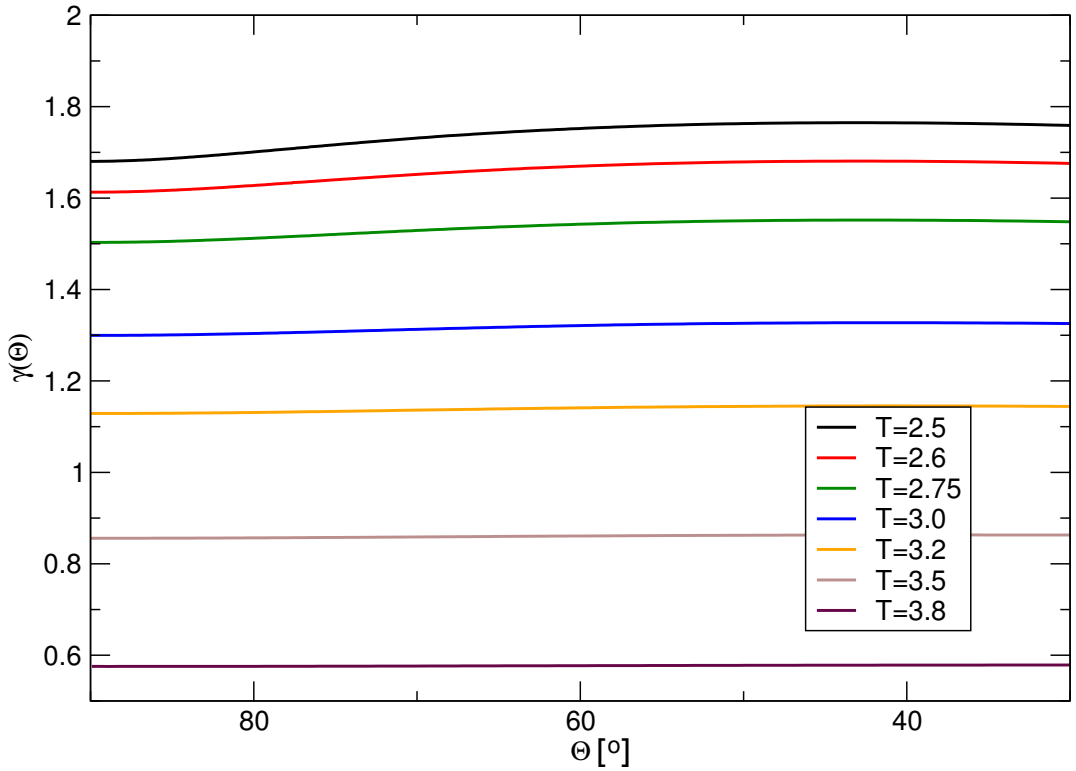


Figure 4.12: $\gamma(\Theta)$ for different temperatures in the range between $T = 2.5$ and $T = 3.8$. This data is obtained via numeric integration according to equation 4.19. The data in figure 4.3 was used for $\gamma(90°)$. As can be seen, the variation in the absolute value of γ is rather small, especially above the roughening temperature T_R. When approaching T_R, the variation increases significantly.

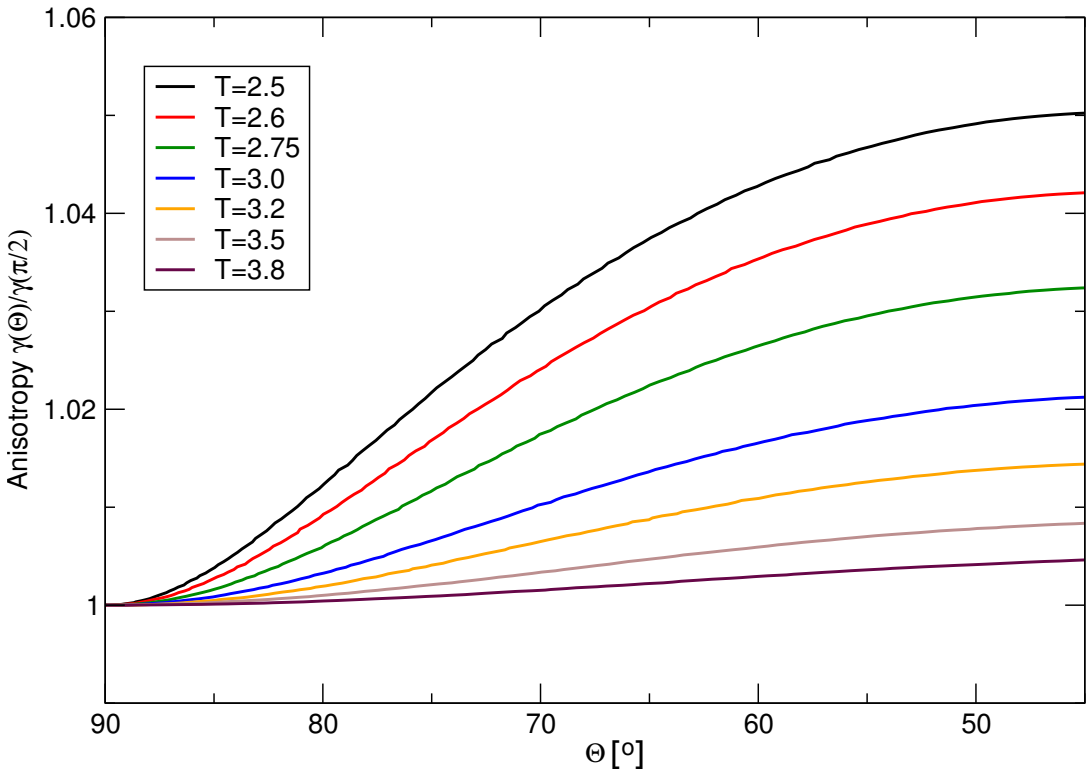


Figure 4.13: Surface anisotropy $\gamma(\Theta)/\gamma(90^\circ)$ for different systems approaching the roughening transition. The data in figure 4.3 was used for $\gamma(90^\circ)$. Lower temperatures feature a significant anisotropy in surface tension which is worth investigating in detail, since it is expected to have an importance in the prediction of nucleation rates and crystal shapes.

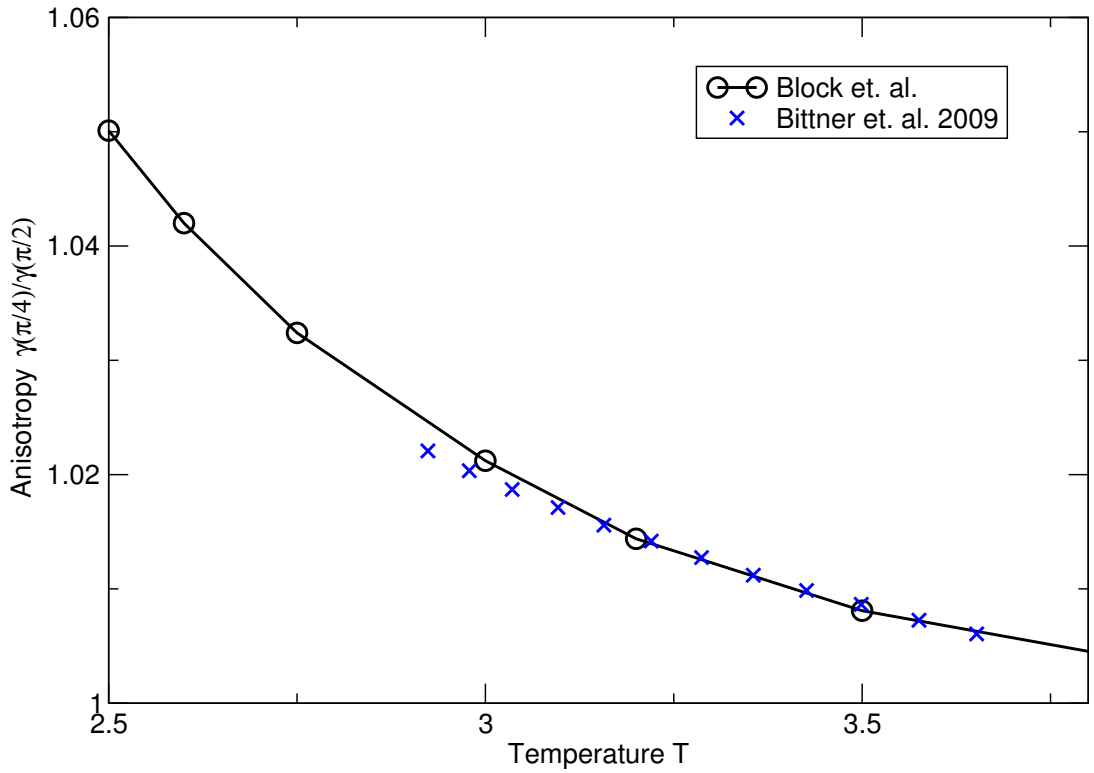


Figure 4.14: Surface anisotropy $\gamma(45°)/\gamma(90°)$ for different temperatures. The data can be compared to [113] and is found to be in good agreement. We are able to predict the value of the anisotropic surface tension for a variety of angles down to the roughening transition temperature T_R. The line only serves as a guide to the eye.

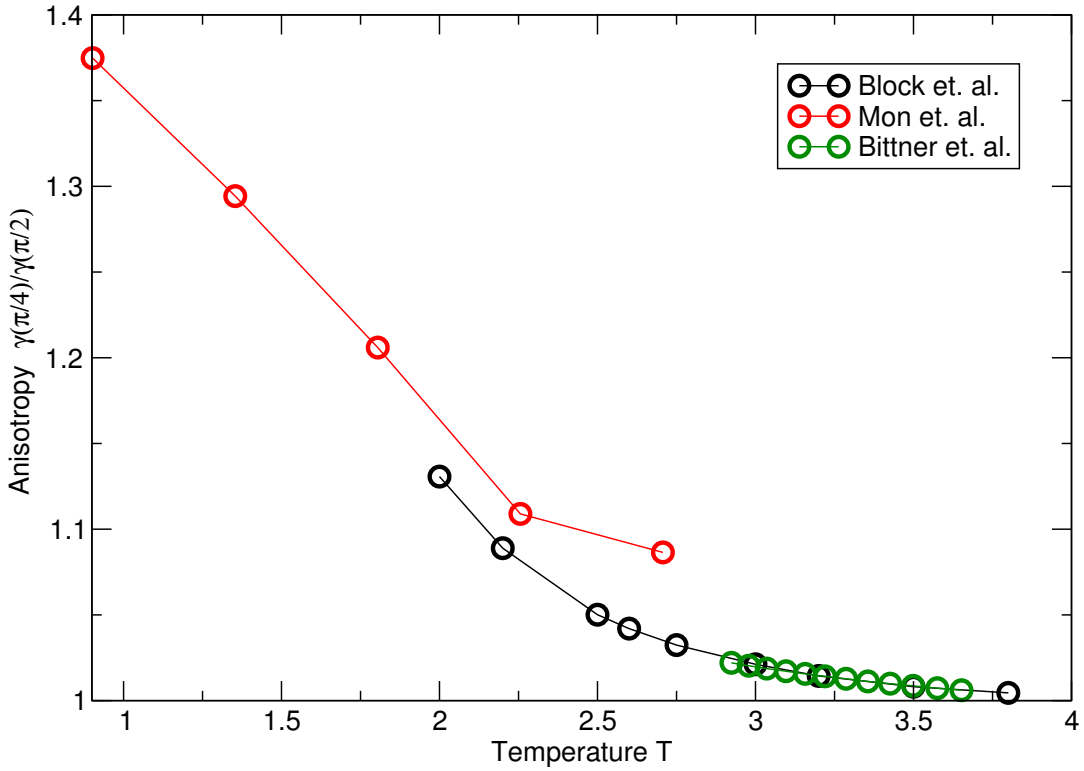


Figure 4.15: Surface anisotropy $\gamma(45°)/\gamma(90°)$ for different temperatures. We also compared our results to the data of Mon et. al. [17]. The publication indicates a significant error in their simulation data, so our results can provide much more accuracy even below T_R. This makes qualitative investigation of the roughening transition feasible and interesting. The quantities that can be analyzed here are the surface stiffness and the *Step Free Energy* as the order parameter of the roughening transition. At lower temperatures ($T < 2.0$) we do not have any data for contact angles of 45°, since the angle is not reached with our simulation range of $H \in [0, 0.8]$. The line only serves as a guide to the eye.

4.6 Surface Stiffness

As elaborated in section 1.3, the surface stiffness is an important property of interfaces, which is characterized by the stiffness coefficient κ. The surface stiffness only has a meaning in the rough phase, because below the roughening transition, the expansion of $\gamma(\Theta)$ has a leading first order term and the stiffness is infinity. For these reasons the analysis has to be restricted to the rough phase where

$$T > T_R.$$

When the contact angle is $\Theta = 90^\circ$, the interface on average coincides with a lattice plane. For small angles we expect

$$\gamma(\Theta) = \gamma(\frac{\pi}{2}) \left[1 + c \left(\frac{\pi}{2} - \Theta\right)^2\right]$$

Note that the interfacial stiffness is defined by

$$\kappa = \gamma\left(\frac{\pi}{2}\right) + \gamma''\left(\Theta = \frac{\pi}{2}\right) = \gamma\left(\frac{\pi}{2}\right)[1 + 2c]$$

So the constant c simply is related to the difference between interfacial stiffness and interfacial tension

$$\kappa/\gamma(\frac{\pi}{2}) - 1 = 2c$$

On the other hand, we can use this expansion in the differential equation for $\gamma(\Theta)$, considering the limit $L_x \to \infty$ and using the abbreviation

$$\Delta\gamma = \gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|)$$

namely

$$\gamma(\Theta)\cos\Theta - \gamma'(\Theta)\sin\Theta = \Delta\gamma$$

$$\gamma'(\Theta) = \frac{\mathrm{d}}{\mathrm{d}\Theta}\gamma(\Theta) = -2c\gamma(\frac{\pi}{2})(\frac{\pi}{2} - \Theta)$$

and hence

$$\left[1 + c\left(\frac{\pi}{2} - \Theta\right)^2\right]\cos\Theta + 2c(\frac{\pi}{2}) - \Theta)\sin\Theta = \Delta\gamma/\gamma(\frac{\pi}{2})$$

since the data of [17] indicates that $\gamma(\Theta) - \gamma(\frac{\pi}{2})$ is small for all Θ, we may expand this equation around the contact angle in the isotropic case (Θ_0).

$$\cos\Theta_0 = \Delta\gamma/\gamma(\frac{\pi}{2})$$

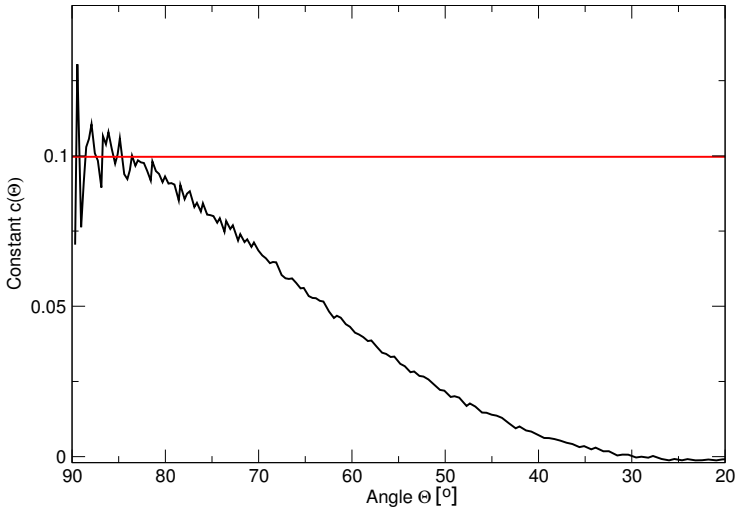


Figure 4.16: The constant c in equation 4.20 can be retrieved from the difference of angles $\Theta - \Theta_0$, which have already been measured. It helps evaluating in which range of angles a linear dependence is justified. This data is for a temperature of $T = 3.0$. The red bar indicates the average value over the interval $[90, 85]$, which can be used as an estimate for c. The data is very noisy and not optimal for the retrieval of a value for c.

to find $[\cos\Theta = \cos\Theta_0 - \sin\Theta_0(\Theta - \Theta_0), \sin\Theta \approx \sin\Theta_0]$ to leading order.

$$\cos\Theta_0 = (\Theta - \Theta_0)\sin\Theta_0 + c(\frac{\pi}{2} - \Theta_0)^2\cos\Theta_0 + 2c(\frac{\pi}{2} - \Theta_0)\sin\Theta_0 = \Delta\gamma/\gamma(\frac{\pi}{2})$$

and hence to leading linear order in c we obtain

$$\Theta - \Theta_0 = c\left[\pi - 2\Theta_0 + \left(\frac{\pi}{2} - \Theta_0\right)^2 / \tan\Theta_0\right]. \tag{4.20}$$

Equation 4.20 we can use to retrieve a value for c (as has been done for $T = 3.0$ in figure 4.16 and for three different tempeatures in figure 4.17). As it turns out at this temperature, a linear dependence of Θ on the angle difference $\Theta - \Theta_0$ is only seen in a small region of angles $[90°, 85°]$. For lower temperatures, this region needs to be even more narrow, for higher temperatures, the region of linear dependence stretches out longer. The values for c obtained from averaging over this range can be found in table 4.18. Of course, it is expected to hold only for small contact angles Θ for which the expansion is still accurate enough.

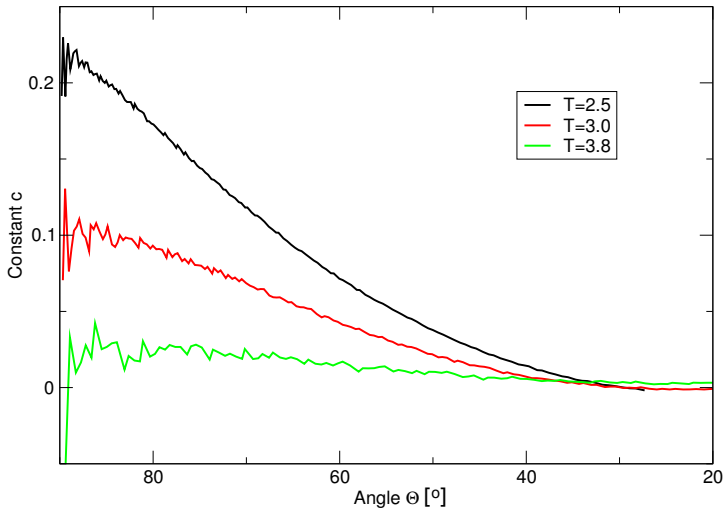


Figure 4.17: The constant c in equation 4.20 for three different temperatures. The range of linear dependence gets more narrow with lower temperatures.

Figure 4.18: The constant c for different temperatures. Unfortunately the method does not produce very satisfying results, especially in the vicinity of the roughening transition.

Temperature T	c	$\kappa/\gamma(90^\circ)$
2.75	0.1404	1.2809
3.0	0.0997	1.1994
3.2	0.0759	1.1518
3.5	0.0471	1.0943
3.8	0.0210	1.0421
4.0	0.0161	1.0322

4.6.1 A Different Approach

The previous method is interesting as a verification and in fact the values obtained are of the expected magnitude, but they lack the accuracy required for a rigorous analysis. We therefore continue the investigation by using a slightly different method to estimate the stiffness coefficient κ. Here, we extract the surface stiffness in the rough phase by a fit of the difference $\gamma(\Theta) - \gamma(90^\circ)$ to a parabola. Since

$$\gamma(\Theta) \approx \gamma(90^\circ) + 1/2\gamma''(90^\circ) \left(\frac{\mathrm{d}h}{\mathrm{d}x}\right)^2 \tag{4.21}$$

and

$$\left(\frac{\mathrm{d}h}{\mathrm{d}x}\right) = cos(\Theta),$$

if we measure the difference

$$\Delta\gamma(\Theta) = \gamma(\Theta) - \gamma(90^\circ), \tag{4.22}$$

we can extract the quadratic contribution

$$\Delta\gamma(\Theta) = \frac{1}{2}\gamma''(\Theta) \cdot (\cos(\Theta))^2$$

and therefore the second derivative $\gamma''(\Theta)$ directly by fitting a parabola in $\cos\Theta$ (see figure 4.19). With a known value for $\gamma(90^\circ)$, the surface stiffness κ can be constructed

$$\rightarrow \kappa/\gamma(90^\circ) = \frac{\gamma(90^\circ) + \gamma''(90^\circ)}{\gamma(90^\circ)}.$$

This procedure can be applied to the data for all temperatures, leading to the data in table 4.1. The fitting range of angles we have used is

$$[90, 85]$$

because here, we can see no significant deviation from the parabola shape (equation 4.21) and the fit is well justified (see figure 4.19). This data can be used to compare our results to the investigation carried out by Hasenbusch et. al. [24]. The comparison is shown in figure 4.20.

4.7 Low Temperatures

The behavior of the interface below T_R is harder to analyze since several different mechanisms are at work here and it is not clear which effects play a role in the qualitative behavior. We have to take a close look at the configurations and

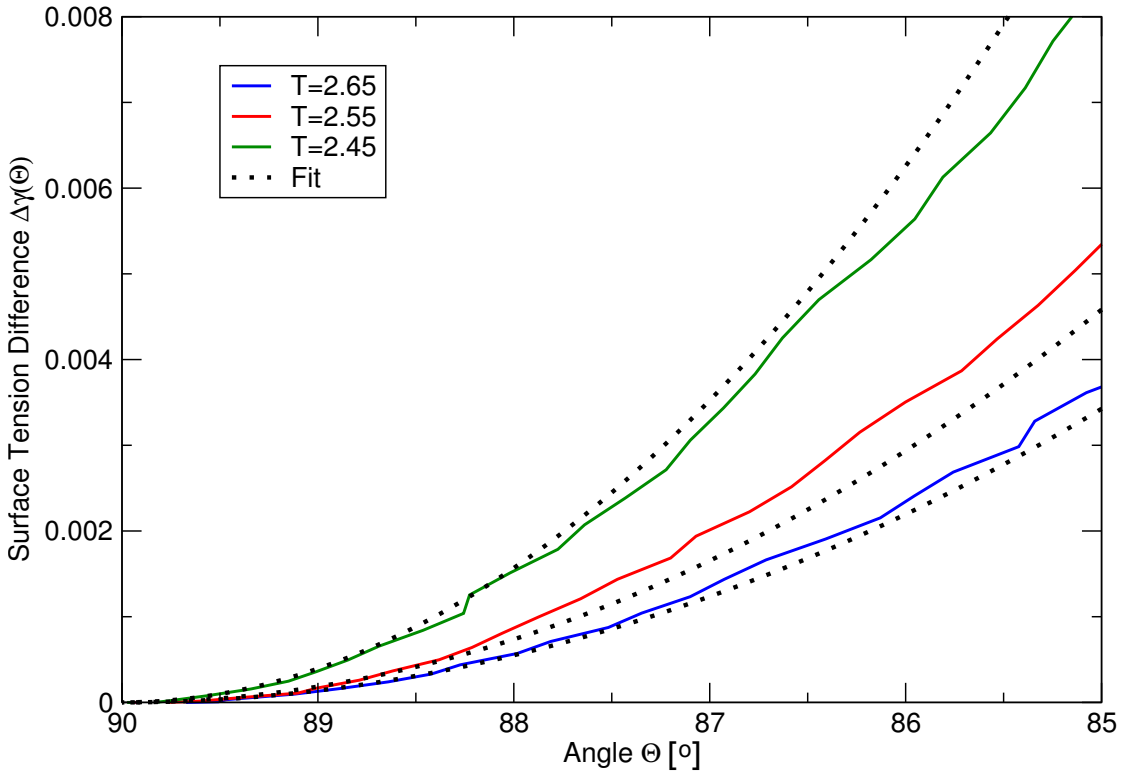


Figure 4.19: Surface tension in the vicinity of $\Theta = 90°$ for different temperatures near the roughening transition at T_R. The curve is fitted to equation 4.6.1 to extract the surface stiffness. The fit needs to be restricted to a small angle around 90°, since the expansion in terms of surface stiffness is only accurate up to leading order.

Temperature T	Tension $\gamma(90°)$[114]	$\gamma''(90°)$	Stiffness $\kappa/\gamma(90°)$
2.5	1.6803	1.7237	2.0258
2.6	1.6130	1.1414	1.7077
2.75	1.5031	0.6418	1.4270
3.0	1.2996	0.2915	1.2242
3.2	1.1289	0.1490	1.1320
3.5	0.8560	0.0562	1.0656
3.8	0.5755	0.0127	1.0221
4.0	0.3962	0.00475244	1.0121
4.2	0.2452	0.00943	0.9615

Table 4.1: Second derivative of the surface tension γ'' for different temperatures. γ'' is extracted with a fit to equation 4.6.1 as shown in figure 4.19. γ'' is needed to construct the surface stiffness $\kappa = \gamma + \gamma''$.

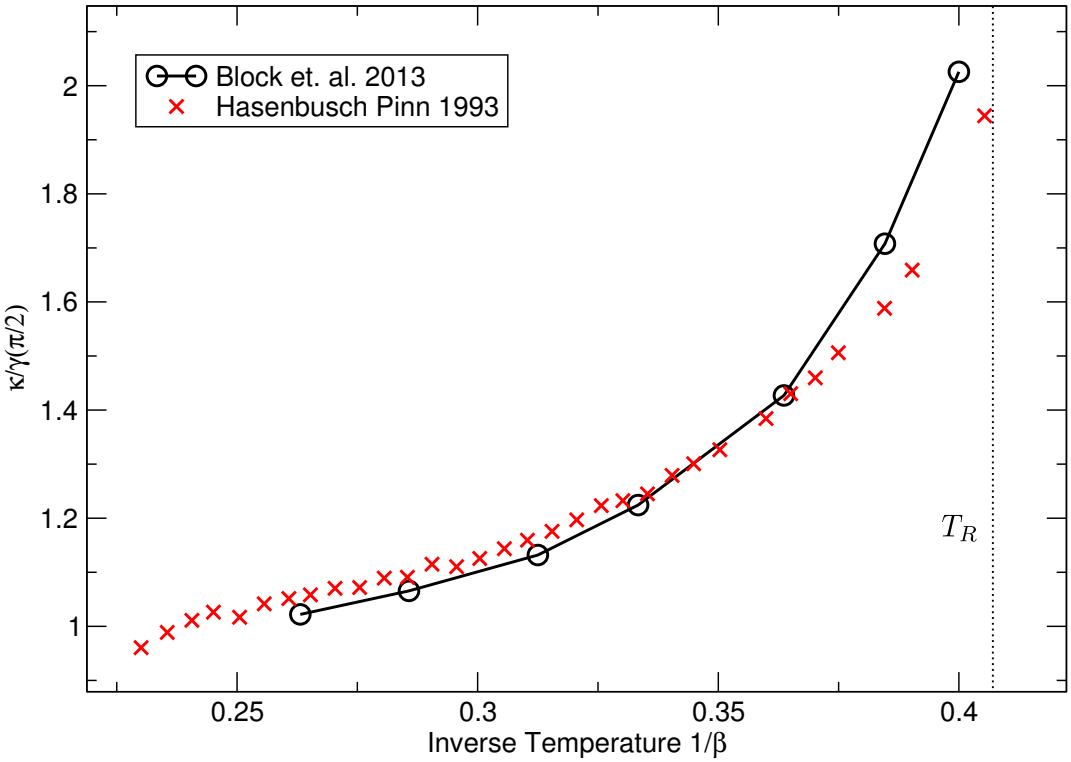


Figure 4.20: Surface Stiffness $\kappa(90°)/\gamma(90°)$ from our analysis in a system of size $88 \times 504 \times 504$ compared to the results obtained by [24]. Our data reflects the theoretical prediction (see figure 1.6), which predicts a vanishing of the surface stiffness at high temperatures and a divergence of the stiffness at temperatures below T_R very well. The line only serves as a guide to the eye.

the small angle behavior of the surface tension and of the contact angle itself at small magnetic fields. At small angles below the roughening temperature, creating a step in the interface costs a specific *Step Free Energy* $f_s(T)$. The smallest angle that an interface below T_R can take on in the case of a lattice of spacing 1 is $\arctan(\frac{1}{L_x})$. Thus the minimum angle that can be formed has a strong dependency on the linear dimension L_x. Indeed, in simulations below the roughening transition (figure 4.21) we see that, for fields below a certain strength H_C, the angle stays practically zero. This effect appears gradually below T_R and gets

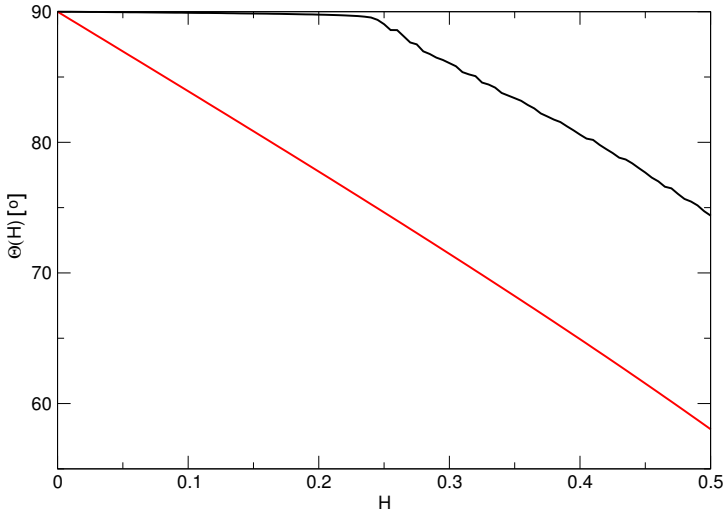


Figure 4.21: Below the roughening transition, the difference between Θ and Θ_0 which has been determined via Young's equation is much less subtle than above the transition. The simulation data is for $T = 2.0$ at a system size of $184 \times 504 \times 504$. The black curve shows the angle calculated from the difference in wall free energies using Young's Equation, while the red line shows the angle Θ calculated from the geometric construction. Below the roughening transition, the system has to reach a critical field H_C before it can form a tilted interface. This critical field is linked to a *Step Free Energy* as explained by the model developed in section 4.7.1.

very strong at low temperatures. Contrary to intuition however, this critical field H_C is independent of the system size L_x. We develop a low temperature model which can explain this behavior. It connects the magnetic field with the inclination angle at a microscopic level and even though it does not capture all visible effects in the simulation data, it can explain the behavior at the critical field H_C and delivers a method to extract the *Step Free Energy* $f_s(T)$ from simulation data.

If the system is at an angle below $\arctan(\frac{1}{L_x})$, only 0 and 1 step configurations are expected to have a significant contribution to the ensemble average, all other configurations are heavily suppressed. The one step configurations can be seen as a single free particle in a one dimensional domain of length L_x.

4.7.1 A Model for Inclination Angles at Low Temperatures

The generalized APBC require

$$m^{(+)}(i, H) = -m^{(-)}(L_x - i, -H)$$

We now assume a three state model in which we only investigate the three lowest energy states (interfaces with 0 and 1 steps along the x-axis). Relative to

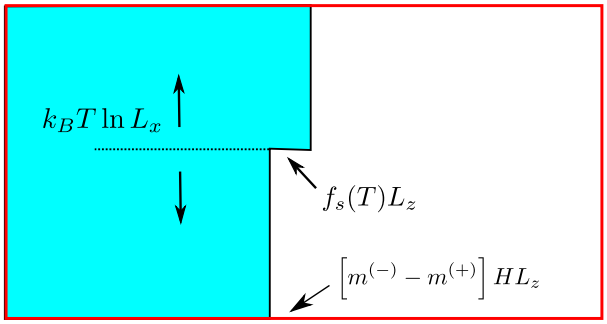


Figure 4.22: The three contributions to the low temperature model of a tilted interface with a single kink illustrated in one picture.

the flat interface perpendicular to the y-axis, we have the following energy costs for inclined interfaces (illustrated in figure 4.22)

$$\Delta E_{+1} = f_s(T) L_z - k_B T \ln L_x - \left[m^{(+)}(L_x, H) - m^{(-)}(L_x, H)\right] H L_z, \tag{4.23}$$

where the first term is the energy gain due to the step along the z-axis, the second term is the step translation entropy along the x-axis, and the last term is the gain due to the boundary field energy, which is roughly ≈ 2 at low temperatures. For a step in the other direction (against the field), we have an energy difference of

$$\Delta E_{-1} = f_s(T) L_z - k_B T \ln L_x + \left[m^{(+)}(L_x, H) - m^{(-)}(L_x, H)\right] H L_z, \tag{4.24}$$

with the same contributions, only that the last term has a contribution in the opposite direction.

In the three-state approximation, the weights of the three states are

$$\begin{aligned} W_0 &= 1 \\ W_{+1} &= L_x \exp\left(\frac{-f_s(T) L_z}{k_B T}\right) \exp\left(\frac{2 H L_z}{k_B T}\right) \\ W_{-1} &= L_x \exp\left(\frac{-f_s(T) L_z}{k_B T}\right) \exp\left(\frac{-2 H L_z}{k_B T}\right) \end{aligned} \tag{4.25}$$

The weights add up to the partition function Z

$$\begin{aligned} Z &= W_0 + W_{+1} + W_{-1} \\ &= 1 + \exp(-\Delta E_+ / k_B T) + \exp(-\Delta E_- / k_B T) \\ &= 1 + 2 L_x \exp\left(\frac{-f_s(T) L_z}{k_B T}\right) \cosh\left(\frac{2 H L_z}{k_B T}\right) \end{aligned} \tag{4.26}$$

which extends to higher order terms if we take more states into consideration. For convenience in the course of this calculation, we define

$$\Phi = \frac{\pi}{2} - \Theta,$$

where Θ is the contact angle as before. By averaging over the value of the respective angle in the discrete angle states that are possible

$$\Phi_i = \tan(i/L_x) \approx i/L_x \tag{4.27}$$

and recalling the definition of the average

$$\langle \Phi \rangle = \frac{1}{Z} \sum_i \Phi_i W_i \tag{4.28}$$

we arrive at the average inclination angle

$$\begin{aligned} \langle \Phi \rangle &= \frac{1}{L_x} \left(W_{+1} - W_{-1} \right) / Z \\ &= \frac{2}{Z} \exp\left(\frac{-f_s(T) L_z}{k_B T} \right) \sinh\left(\frac{2HL_z}{k_B T} \right) \end{aligned} \tag{4.29}$$

We have to be careful to linearize this expression with respect to $\frac{2HL_z}{k_B T}$, since for large linear z-dimensions, the assumption $\frac{2HL_z}{k_B T} \ll 1$ breaks down already for very small H.

For large enough $\frac{2HL_z}{k_B T}$, it is clear that the state $+1$ must dominate while for $H = 0$ the 0 state must dominate. It is worthwhile to have a look at where the transition happens. To this end, assume that $\frac{2HL_z}{k_B T} \gg 1$, so $\sinh\left(\frac{2HL_z}{k_B T}\right) = \frac{1}{2} \exp\left(\frac{2HL_z}{k_B T}\right)$, hence

$$\langle \Phi \rangle \approx \frac{\exp\left(\frac{-f_s(T) - 2H}{k_B T} L_z \right)}{1 + L_x \exp\left(\frac{-f_s(T) - 2H}{k_B T} L_z \right)} \tag{4.30}$$

If

$$\exp\left(\frac{-f_s(T) - 2H}{k_B T} L_z \right) \gg 1/L_x, \tag{4.31}$$

then $Z \approx L_x \exp\left(\frac{-f_s(T) - 2H}{k_B T} L_z \right)$ and thus

$$\langle \Phi \rangle \approx \frac{1}{L_x}$$

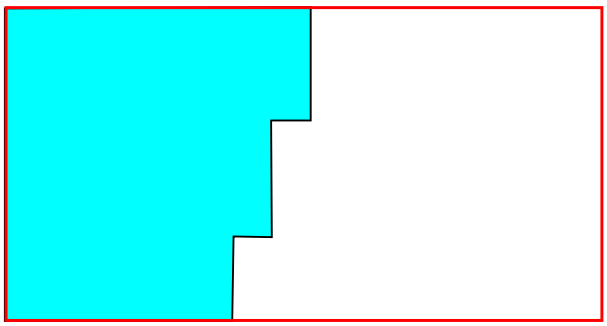


Figure 4.23: For larger inclination angles and larger external fields, the model needs to incorporate multiple step configurations as well.

taking equation 4.31 as a criterion for the crossover between the two states, yields a critical field H_C, where a crossover between a *majority o* and a *majority +1* state happens.

$$H_C = \frac{1}{2} f_s(T) - \frac{k_B T}{2 L_z \ln L_x} \tag{4.32}$$

and thus

$$H_C \to \frac{1}{2} f_s(T) \tag{4.33}$$

for large enough systems. In our case ($L_x = 184, L_z = 504$), the second contribution takes on a numerical value of

$$\frac{k_B T}{2 L_z \ln L_x} \approx 0.0003804$$

which can safely be neglected from our calculations, and we can set $H_C = \frac{1}{2} f_s(T)$ within our numerical accuracy.

This simple model breaks down as soon as there is a significant contribution of the $+1$, because then, states with more than one kink will be occupied, and we have to take those states into account that have more than one kink along the surface (figure 4.23). The energy of these states is

$$\begin{aligned}
\Delta E_{+2} &= 2 f_s(T) L_z - 2 k_B T \ln(L_x/2) - 4 H L_z \\
\Delta E_{-2} &= 2 f_s(T) L_z - 2 k_B T \ln(L_x/2) + 4 H L_z \\
\Delta E_{+3} &= 3 f_s(T) L_z - 3 k_B T \ln(L_x/2) - 6 H L_z \\
\Delta E_{-3} &= 3 f_s(T) L_z - 3 k_B T \ln(L_x/2) + 6 H L_z \\
&\ldots
\end{aligned} \tag{4.34}$$

To generalize the approach above, we deduce the general formula for the weight of a state with n kinks

$$\begin{aligned} W_{+n} &= \left(\frac{L_x}{n}\right)^n \exp\left(-n\left[\frac{f_s(T)-2H}{k_BT}\right]L_z\right) \\ W_{-n} &= \left(\frac{L_x}{n}\right)^n \exp\left(-n\left[\frac{f_s(T)+2H}{k_BT}\right]L_z\right) \end{aligned} \tag{4.35}$$

for $n \in [1, 2, 3...]$.

Using equation 4.27 for the angle in a state with n steps and the definition of the average, we can calculate the average angle at a given field

$$\langle\Phi\rangle = \frac{\sum_{i=1}^{L_x} i/L_x \left[W_{+n} - W_{-n}\right]}{1 + \sum_{i=1}^{L_x}\left[W_{+n} + W_{-n}\right]} \tag{4.36}$$

The result for this calculation is shown in figure 4.24. The average angle stays zero until it reaches the critical field H_C, at which it jumps to a large value. The value is 60° in our analysis even though this angle does not have any meaning because the simplifications fail at this large angle. The step is very sharp so that even at smaller system sizes, it can already be treated as a step. From this step on, all other higher angle states are instantly available and are no longer suppressed which provides an explanation why the crossover field H_C does not have a large L_x dependency. It is important to note that even though the angles of the individual states are heavily system size dependent the total average is not, because as soon as the first angle state starts to fill, pretty much all angle states are available to be filled.

Our data for $T = 2.0$ shows that in simulations, the step is not as sharp as predicted by the low temperature model (see figure 4.25). At angles below the critical field the angle already indicates a very small population of non-zero angle states. At angles above the critical field, the rise in magnitude is not as rapid as predicted. When approaching even lower temperatures the shape of $\Theta(H)$ starts to resemble the theory curves (see figure 4.26). Despite the discrepancies to the low temperature model, it prooves useful to extract an estimate of the *Step Free Energy*. The theory curve for a specific critical field is compared to the simulation data so that the step of both curves match. Repeating this procedure for various temperatures approaching T_R from below allows the investigation of the scaling behavior of the *Step Free Energy* in the next section. The error in this procedure is estimated to be about the width of the theoretical step.

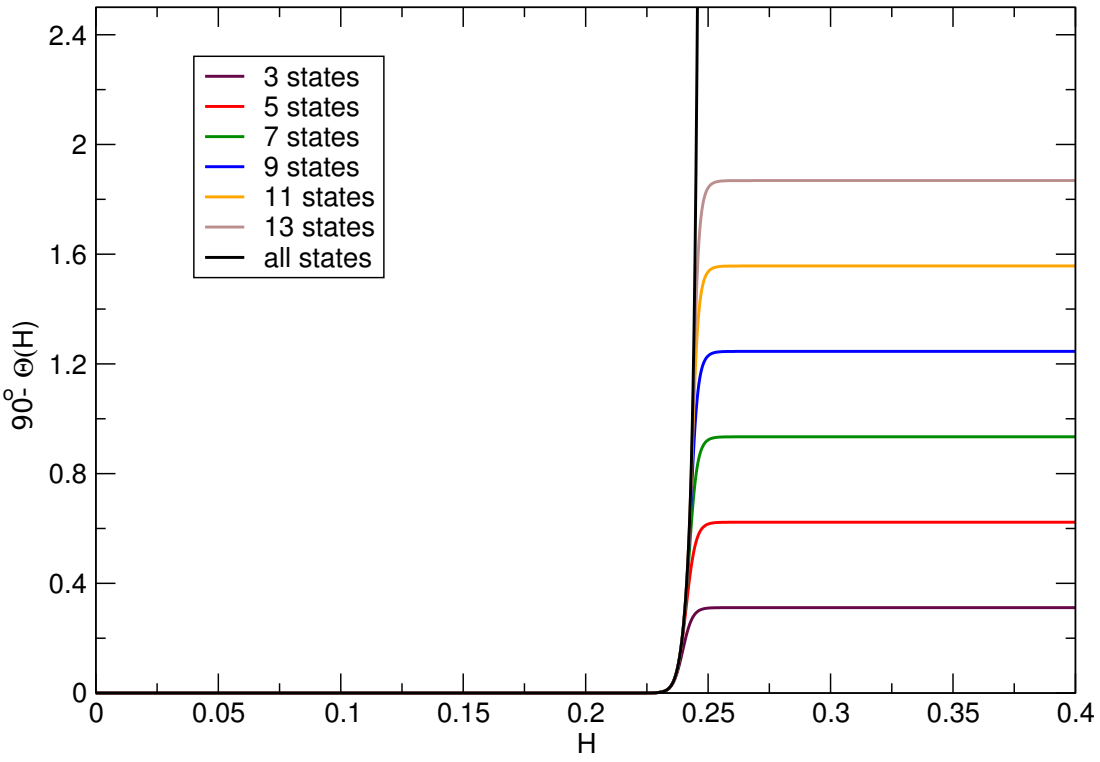


Figure 4.24: An illustration of the angle expectation value $\langle \Phi \rangle$ for $f_s = 0.5$ and $T = 2.0$. Taking the occupation of higher order states into account, the saturation angle is much higher, but the behavior in the onset of population in the non-zero angle states does not change. The transition is very sharp and is accurate only in the very low temperature regime.

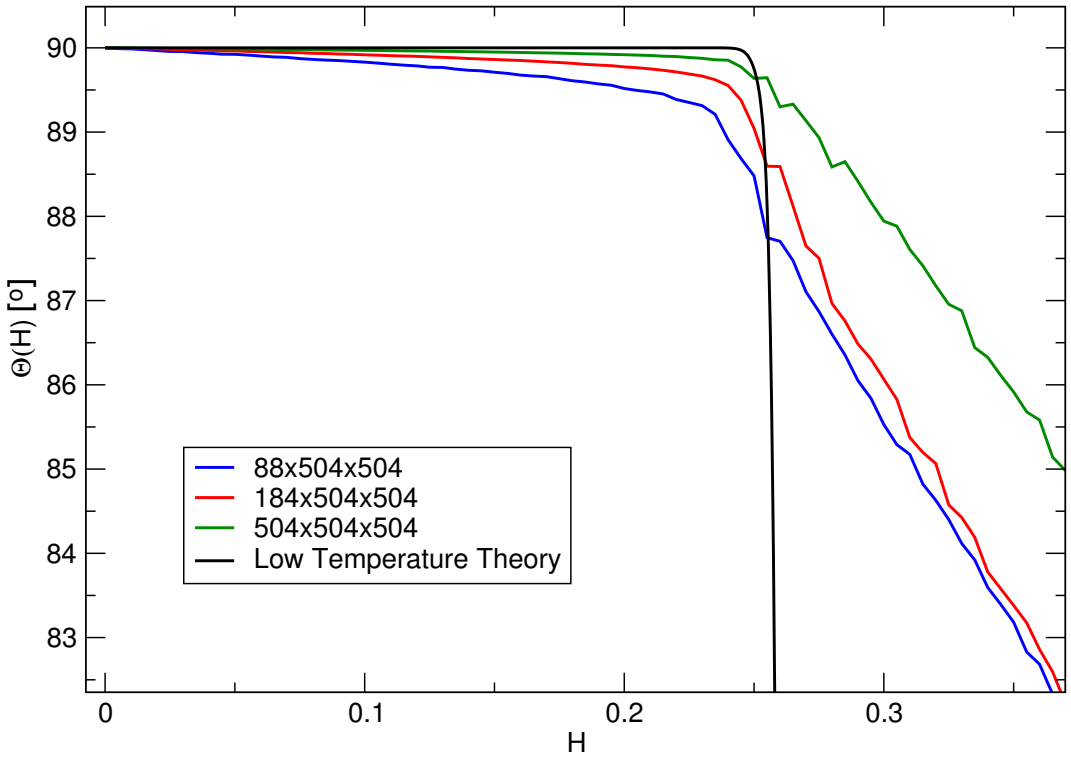


Figure 4.25: The contact angle data at $T = 2.0$ for different system sizes L_x in comparison with the theory curve for T=2.0 and a *Step Free Energy* of $f_s = 0.52$. The behavior up to the critical field H_C indicates agreement with our theory in the limit of an infinite system. The large field behavior however is not captured (figure 4.24), which is believed to be governed by more complex interactions such as effective kink-kink repulsions which modify the large field behavior. Nevertheless, the crossover position can be used to estimate the *Step Free Energy* and compare it to literature [17].

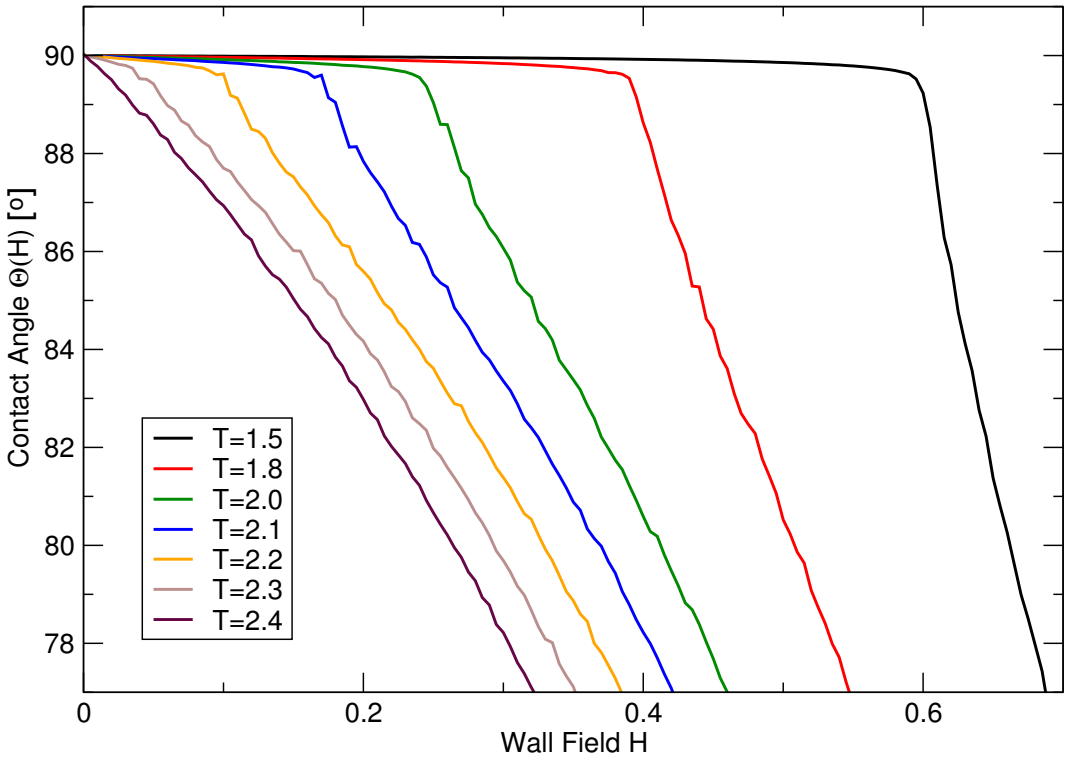


Figure 4.26: At low temperatures, the contact angle curve resembles more and more to that of the low temperature model since the step at H_C gets sharper with lower temperatures.

4.8 Scaling Behavior

The procedure to estimate the *Step Free Energy* in section 4.7.1 is repeated for several temperatures between $T = 1.5$ and $T = 2.4$ (figure 4.27). The temperature dependence allows for a comparison with the theoretically predicted [25, 115] non-universal scaling behavior of the *Step Free Energy* $f_s(T)$ below T_R

$$\frac{T}{f_s(T)} \sim \exp\left(\frac{\pi}{2c\sqrt{\frac{T_R - T}{T_R}}}\right) \quad , \quad T < T_R \tag{4.37}$$

where c is a constant which we determine via a least-squares fit from the curve in figure 4.27 as

$$c = 1.94 \pm 0.17.$$

References [17, 25] estimated c to be

$$c = 1.57 \pm 0.07$$

for the (100) plane of a simple cubic lattice. On the other hand, the analysis of the surfaces stiffness κ in [25] suggests a non-universal scaling behavior of

$$\beta\kappa(T) = \frac{\pi}{2}\left(1 - c \cdot \sqrt{\frac{T - T_R}{T_C}}\right) \quad , \quad T > T_R \tag{4.38}$$

close but above the roughening temperature. At the roughening transition temperature T_R, $\beta\kappa$ takes on the value $\frac{\pi}{2}$ and jumps to ∞ right below it. Near to T_R, the second derivative $\gamma''(T)$ is harder to estimate according to the procedure explained in section 4.6.1, because of increasing fluctuations. The outcome of the parabola fit becomes very sensitive to the choice of the fitting range for the parabola (see figure 4.28). For a small amount of data points, the data becomes very noisy. For a larger amount of data points, higher order effects are expected to overtake the critical behavior. As a tradeoff, we choose 10 data points for the analysis of the critical behavior. The prediction (equation 4.38) is compared to the result of the fitting procedure with 10 data points in figure 4.29. Even though a large amount of statistics was needed to produce meaningful measurements, the computational effort could be handled by our parallel GPU implementation. Our simulation results are (considering the statistical error that can be estimated from figure 4.28) in agreement with the theoretical prediction and show better agreement in the proximity of T_R than the results by [114].

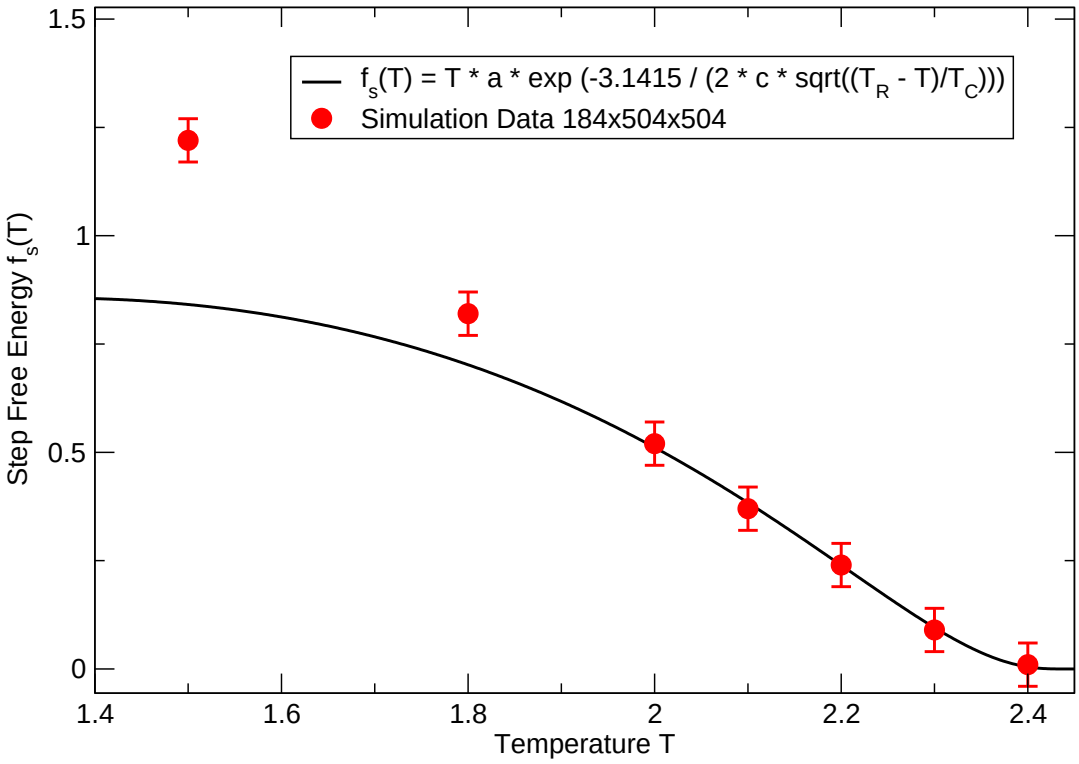


Figure 4.27: The *Step Free Energy* in dependence of the temperature T below the roughening transition T_R shows good agreement with the theoretically predicted non-universal scaling behavior. The error bars approximate the width of the step at H_C. The data points at $T = 1.5$ and $T = 1.8$ are excluded from the least squares fit.

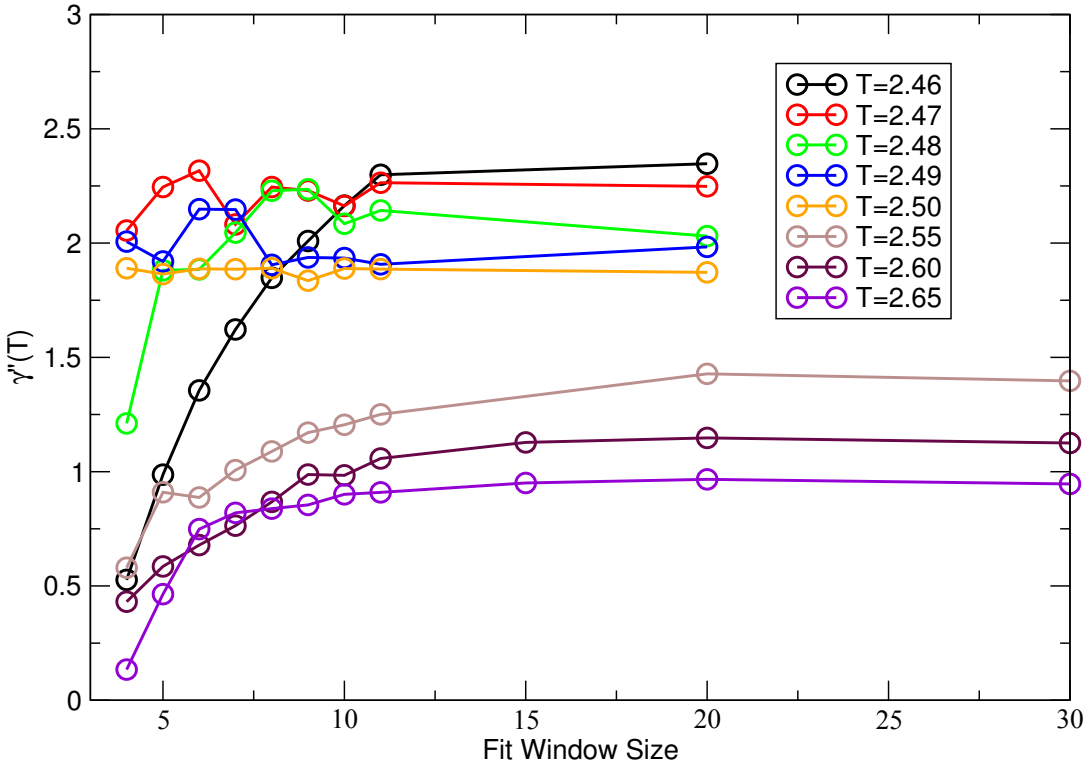


Figure 4.28: The second derivative $\gamma''(T)$ in dependence on the number of data points included in the fit (section 4.6.1). For the thermodynamic integration, the distance between two data points is $\Delta H = 0.005$. The connecting lines between the data points only serve as a guide to the eye.

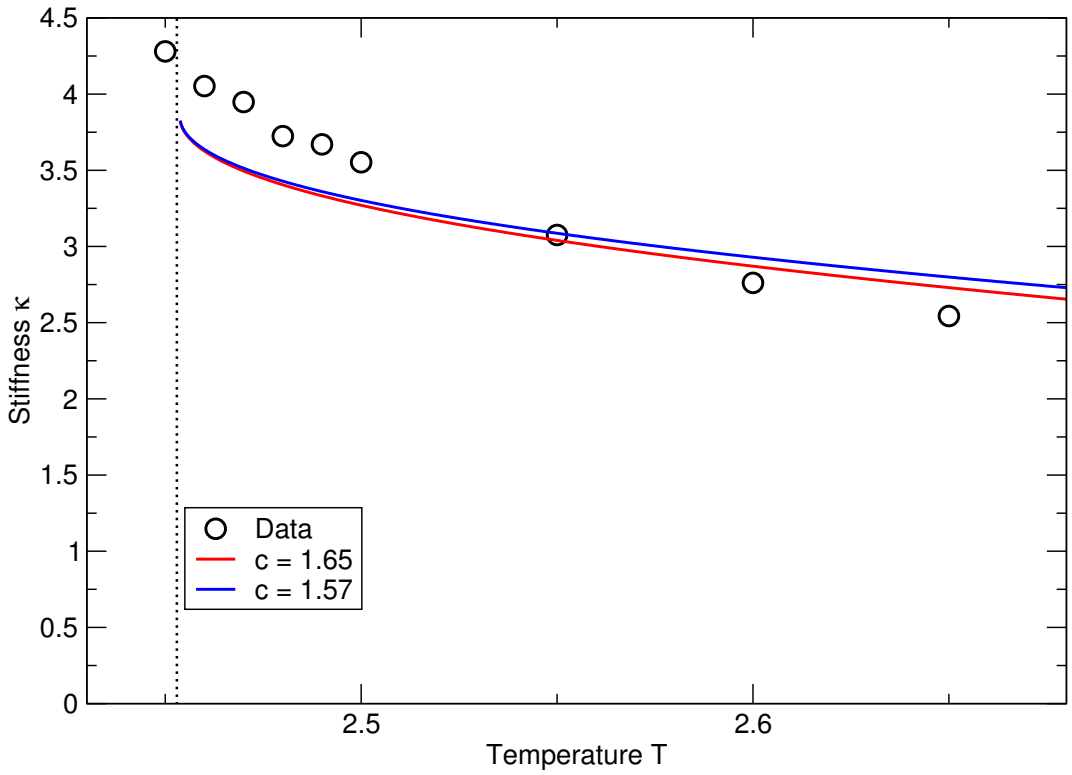


Figure 4.29: The simulation data in a system of size $184 \times 504 \times 504$ in comparison with the exact theoretical prediction $\beta\kappa(T) = \frac{\pi}{2}\left(1 - c \cdot \sqrt{\frac{T-T_R}{T_C}}\right)$. For the blue line, c is chosen according to the analysis of [17, 25], while the red line shows the same scaling law with c choosen according to the outcome of our own analysis of the critical scaling of the *Step Free Energy*.

4.9 Line Tension Contribution

The line tension contribution has so far not been treated and needs to be added as an additive factor that increases with the size of the system in the L_z dimension. Backtracking and looking again at equation 4.1, we had the following equation to sum up the total interfacial free energy of the system:

$$F_{\text{int}} = 2L_z\tau(H,\Theta) + \tilde{L}L_z\gamma(\Theta) + L_z\frac{L_y + \tilde{L}\cos\Theta}{2}\gamma^{(+)}(+|H|)$$
$$+L_z\left(\frac{L_y - \tilde{L}\cos\Theta}{2}\right)\gamma^{(-)} + L_z\left(\frac{L_y - \tilde{L}\cos\Theta}{2}\right)\gamma^{(+)}(-|H|)$$
$$+L_z\left(\frac{L_y + \tilde{L}\cos\Theta}{2}\right)\gamma^{(-)}(-|H|)$$

using the symmetries 4.2 plus the additional symmetry property

$$\tau(+|H|,\Theta) = \tau(-|H|,\Theta)$$

as well as $\tilde{L} = L_x/\sin\Theta$:

$$F_{\text{int}} = 2L_z\tau(H,\Theta) + L_xL_z\gamma(\Theta)/\sin\Theta + L_z(L_y + L_x/\tan\Theta)\gamma^{(+)}(+|H|)$$
$$+L_z(L_y - L_x/\tan\Theta)\gamma^{(+)}(-|H|)$$

To find the contact angle, F_{int} has to be minimized at fixed H with respect to Θ. As discussed above in section 4.4.1, we once again minimize the surface free energy $F_{\text{int}}(\Theta)$ (equation 4.4) with respect to Θ:

$$\left(\frac{\partial F_{\text{int}}}{\partial\Theta}\right)_{H,T} = 0$$

This time we also assume that the partial derivation $\frac{\partial\tau}{\partial\Theta}_H \neq 0$, so we introduce further terms to the expression 4.18

$$\begin{aligned} 0 &= 2L_z\left(\frac{\partial\tau}{\partial\Theta}\right)_H \\ &+ \frac{1}{\sin^2\Theta}\cdot L_xL_z\left[\left(\gamma'(\Theta)\sin\Theta - \gamma(\Theta)\cos\Theta\right) + \right. \\ &+ \left.\left(\gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|)\right)\right] \end{aligned} \tag{4.39}$$

Therefore we arrive at yet another modified Young's Equation which includes a term that comes from the line tension τ

$$\gamma(\Theta)\cos\Theta - \gamma'(\Theta)\sin\Theta - 2\sin^2\Theta\left(\frac{\partial\tau}{\partial\Theta}\right)_H\frac{1}{L_x} = \gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|) \tag{4.40}$$

In the limit $L_x \to \infty$, the line tension correction becomes negligible, and the previous form is obtained:

$$\gamma(\Theta)\cos\Theta - \gamma'(\Theta)\sin\Theta = -\frac{\mathrm{d}}{\mathrm{d}\Theta}\left(\gamma(\Theta)\sin\Theta\right) = \gamma^{(+)}(-|H|) - \gamma^{(+)}(+|H|) \tag{4.41}$$

The term involving $(\partial\tau/\partial\Theta)_H$ must not be neglected for finite L_x.

We solve equation 4.41 for the derivative $\gamma'(H)$ the same way as before, only with the additional $(\partial\tau/\partial\Theta)_H$ term

$$\sin\Theta\gamma'(\Theta) + \Delta\gamma = \frac{2\sin^2\Theta}{L_x}\tau'(\Theta) - \gamma(\Theta)\cos\Theta$$

$$\sin\Theta\gamma'(\Theta) = \frac{2\sin^2\Theta}{L_x}\tau'(\Theta) - \gamma(\Theta)\cos\Theta - \Delta\gamma$$

$$\Rightarrow \gamma'(\Theta) = \frac{2}{L_x}\sin\Theta\tau'(\Theta) - \gamma(\Theta)\frac{\cos\Theta}{\sin\Theta} - \frac{\Delta\gamma}{\sin\Theta}$$

which is the same differential equation as before, just with an additional τ' term, which disappears in the limit $L_x \to \infty$, where the first term can be neglected for large enough systems. Since $\gamma'(\Theta)$ can be very small, when the interface tension is almost orientation-independent, the term involving $(\partial\tau/\partial\Theta)_H$ must not be neglected for finite L_x. This contribution can be determined by measuring the surface free energy above $F_{\mathrm{s,\,int}}$ for different system sizes and building free energy differences between them. By keeping the products L_xL_z and L_yL_z constant, we can make sure that the above surface contributions to $F_{\mathrm{s,\,int}}$ cancel out, and only the line contribution is left. This lead to technical difficulties, because the GPU simulation is only suited for certain system sizes. By sacrificing some speed over flexibility, we managed to bring down the system size steps down to 8 in x and y dimension and 16 in z dimension. The first system simulated is the following

$$L_x = 24, L_y = 144, L_z = 64$$

when varying L_x, this leads to four other system sizes that can be used while keeping the products L_xL_z and L_yL_z constant:

L_x	L_y	L_z	L_xL_z	L_yL_z
16	96	96	1536	9216
24	144	64	1536	9216
32	192	48	1536	9216
48	288	32	1536	9216
96	576	16	1536	9216

This allows us to simulate the following small systems and calculating the free energy difference by thermodynamic integration. The simulation method needs to be adjusted. The variation in the free energy is really small, and the only contribution left can be attributed to a line tension in the system. When looking at the variation between two of these system sizes (L_y^1 and L_y^2), the difference in free energy needs to be of the form

$$\frac{\Delta F_{\text{s, int}}^{\Delta L_y}}{\Delta L_y} = \frac{\Delta F_{\text{s, int}}^{L_y^1} - \Delta F_{\text{s, int}}^{L_y^2}}{L_y^1 - L_y^2}$$

where the Δ denotes a difference from $H = 0$:

$$\Delta F_{\text{s, int}}(H) = F_{\text{s, int}}(H) - F_{\text{s, int}}(H = 0)$$

A line tension contribution to this equation has to be of the form

$$\Delta F_{\text{s, int}}^{\Delta L_y} = 2 \cdot \Delta L_y \left(\tau(H) - \tau(0)\right) \tag{4.42}$$

because there are two lines of length L_z contributing to the free energy. This means a linear fit to the data should reveal the line tension as the slope of the linear regression fit.

A linear fit to a function

$$f(x) = a + b \cdot x$$

will reveal $b = 2\Delta\tau$ which is the difference in line tension from a flat interface, so the anisotropy of the line tension can be evaluated.

Fitting data at $T = 3.0$ with $H = 0.15, H = 0.30$ and $H = 0.45$ is shown in figure 4.30. For large enough linear dimension L_z the free energy difference shows indeed a linear dependence. For small sizes in L_z

$$96 \times 576 \times 16 \quad \text{and} \quad 48 \times 288 \times 32$$

the data suffers from finite-size effects, so the data for small sizes is not included in the analysis. The simulation data of the three system sizes

$$16 \times 96 \times 96 \quad 24 \times 144 \times 64 \quad 32 \times 192 \times 48$$

is used for a linear fit. This allows extraction of the slope b and in turn of the line tension difference $\Delta\tau$. The line tension difference is added to the values for τ that have been estimated by Kim et al. [42]. The resulting angle dependent line tension $\tau(\Theta)$ is depicted in figure 4.31. For better visualization of the temperature dependence, the same quantity is plotted for four different constant contact angles in figure 4.32. We observe an increase of the negative line tension contribution for angles smaller than 90° for temperatures above $T = 2.75$ while for even lower temperatures, the absolute value of the line tension increases even further until an angle of about 70° before decreasing again as for larger temperatures.

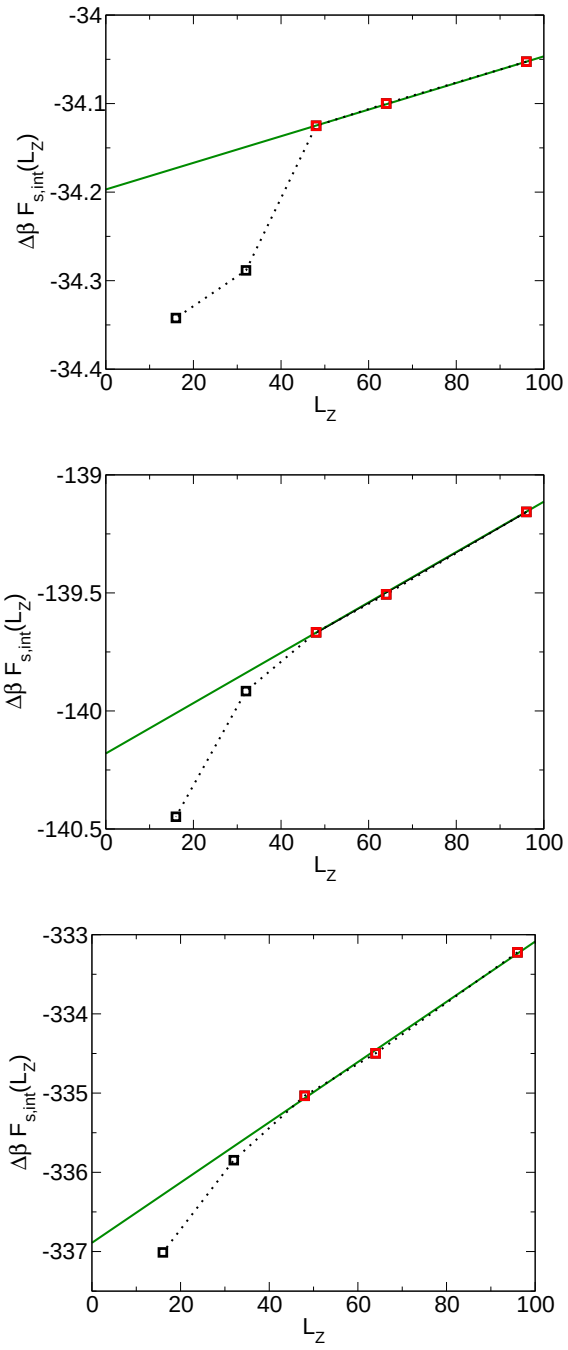


Figure 4.30: The free energy differences $\Delta F_{\mathrm{s,int}}(L_z)$ are expected to have only a linear dependence on the linear dimension L_z, if all system sizes are varied so that the products $L_x L_z$ and $L_y L_z$ are kept constant. All graphs are shown for a system of temperature $T = 3.0$. The uppermost graph shows the dependence for $H = 0.15$, the middle one for $H = 0.35$ and the lowermost for $H = 0.45$.

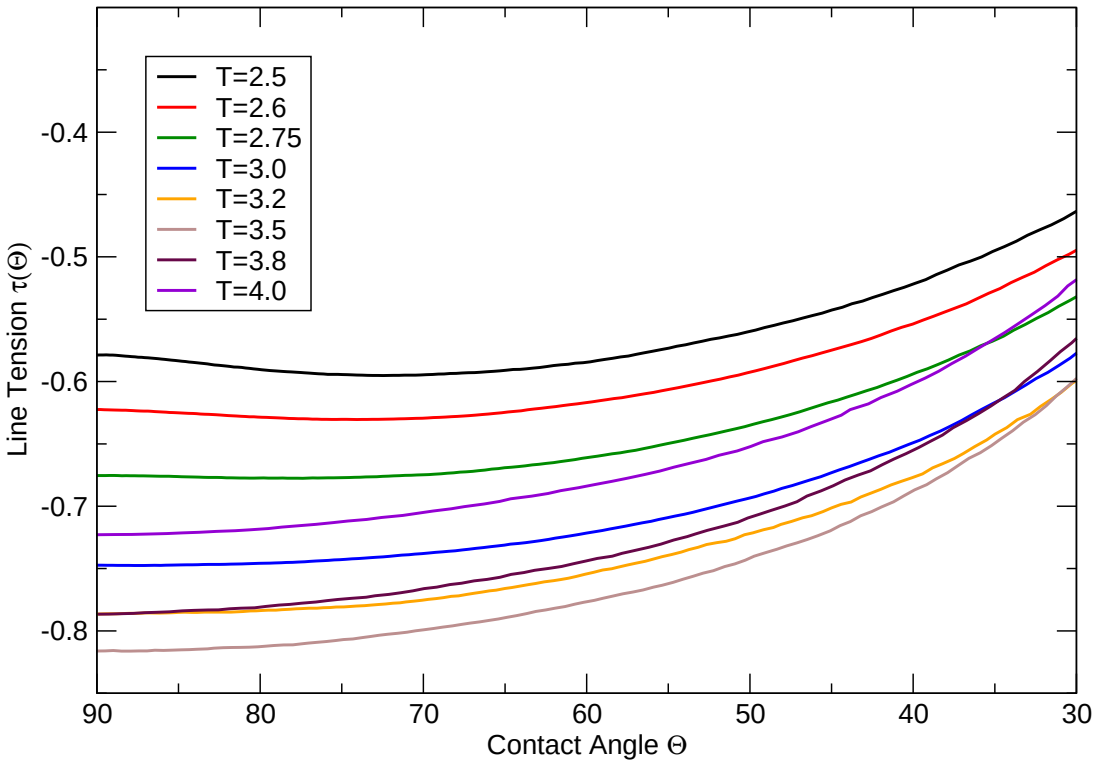


Figure 4.31: Anisotropic line tension for different temperatures ranging from $T = 2.5$ up to $T = 4.0$. The most important observation is that the line tension decreases for interface orientations that are not parallel to the lattice plane.

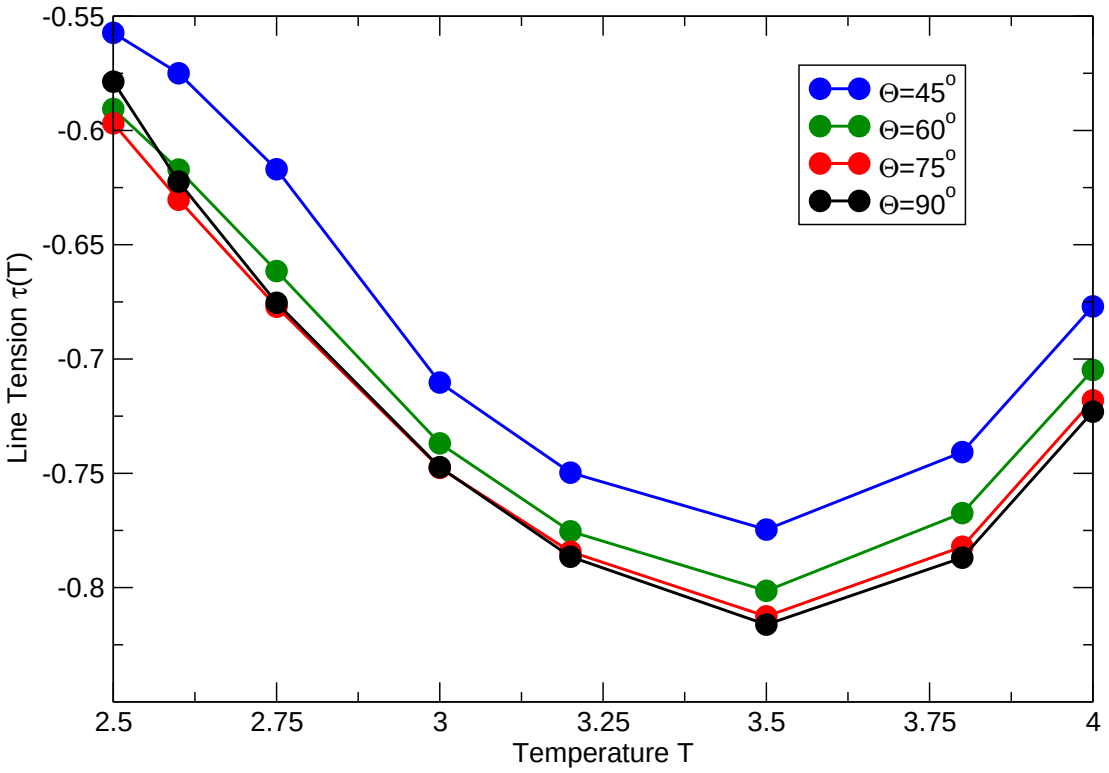


Figure 4.32: Line Tension temperature dependence for different angles $\Theta = 75°$, $\Theta = 60°$ and $\Theta = 45°$. The maximal anisotropy $\frac{\tau(\Theta)}{\tau(90°)}$ is around $T = 3.5$. This is also the temperature of the lowest absolute line tension $\tau(90°)$. The lines between the data points only serve as a guide to the eye.

Chapter 5

A Finite Volume Evolution Galerkin Method on the GPU

5.1 Introduction

Monte Carlo simulations in physical systems such as the Ising model are not the only applications that can benefit greatly from GPU acceleration. The Ising model serves as a very good test case since it is very simple to implement. At the same time it allows to gather interesting physical insights as shown in previous chapters. Another field in which we successfully used GPU acceleration during my PhD thesis is a novel multidimensional Discontinuous Galerkin flux operator to simulate physical flows. In [116], new large time step Finite Volume Evolution Galerkin (FVEG) methods for geophysical flows where proposed. They combine the simplicity of *Finite Volume methods* with the theory of bicharacteristics yielding a genuinely multidimensional finite volume scheme. The Evolution Galerkin operator which is used in order to evaluate fluxes over the cell interfaces can be interpreted as a multidimensional approximate numerical flux function [117, 118]. The simulations we carried out confirm high efficiency and good multidimensional resolution of the scheme. In [116], the FVEG scheme has been considered only on regular rectangular meshes. The approach used in our work uses the scheme of [117] for adaptive irregular meshes in order to allow more efficient simulations of various localized flow structures.

My work involved only the parallelization of the actual evolution operator and porting it to the GPU using NVidia's CUDA framework. It has been published in [110]. This chapter details the content of this publication. The wording of the present chapter is in large parts almost identical to this publication with the exception of this introduction and more detailed descriptions.

Using the GPU allowed the computer simulations to run much faster. We benchmarked the GPU accelerated code on regular and adaptive grids and compared the results with CPU simulations revealing good agreement in accuracy.

For spatial approximation we use second order polynomials with h-adaptive

mesh refinement method and the explicit third order Runge-Kutta method for time integration. For the GPU accelerated parts of the code we have obtained a significant speedup of a factor up to 30 in comparison to a single CPU core, with a potential for further improvement of the performance of the code. Still unfinished is the implementation of an implicit time integration method on the GPU.

The chapter is organized as follows: First, the concepts needed to describe a geophysical flow are introduced, the methods for solving the governing partial differential equations are elaborated on. The multidimensional evolution operator used for flux integration along cell interfaces is described in Section 5.8. In our numerical experiments discussed in Section 5.11, we benchmark the new GPU accelerated code on regular and adaptive grids, using quadratic polynomials for spatial approximation and an explicit third order Runge-Kutta scheme for time integration. In Section 5.12 we will discuss implementation of the evolution operator on the GPU. A recently released paper by Yelash et. al. [119] compares different time integration schemes. In particular, semi-implicit schemes are used in order to overcome the strong stability condition for time steps given by the Courant-Friedrichs-Lewy (CFL) number CFL $= \frac{u\Delta t}{\Delta x}$.

5.2 Notations in Fluid Dynamics

To describe a fluid, a few essential symbols and names for the properties of the fluid need to be introduced to facilitate the use of these properties later.

$\Omega \in R^2$	domain occupied by the fluid
$\mathbf{u} = (u, v)^T$	velocity vector
ρ	density
p	pressure
θ	potential temperature
g	gravitational constant
c_p	isobaric specific heat capacity
c_v	isochoric specific heat capacity
$\gamma = c_p / c_v$	adiabatic constant
$R = c_p - c_v$	gas constant
$\partial_t = \frac{\partial}{\partial t}$	partial derivative with respect to the time t

5.3 Partial Differential Equation

The system of equations we will be dealing with is a time dependent partial differential equation in two spacial dimensions x and y of the form

$$\partial_t \mathbf{w} + \mathbf{A}(\mathbf{w})\nabla\mathbf{w} + \mathbf{B}(\mathbf{w}) = 0 \tag{5.1}$$

where $\mathbf{w}$ is a solution vector of dimensionality m, and $\mathbf{A}, \mathbf{B}$ are $m \times m$ matrices.

The equation system 5.1 is called is *quasi-linear* if $\mathbf{B}(\mathbf{w})$ depends only linearly on $\mathbf{w}$ and $\mathbf{A}(\mathbf{w})$ is a function of the solution vector $\mathbf{w}$. Likewise, 5.1 is called *hyperbolic*, if $\mathbf{A}$ has m real eigenvalues $\lambda_1, ..., \lambda_m$ with a corresponding set of m linearly independent eigenvectors $\mathbf{K}^{(1)}, ..., \mathbf{K}^{(m)}$. A hyperbolic system with $\mathbf{B}(\mathbf{w}) = 0$ is called a hyperbolic conservation law.

5.4 Conservation Laws in Inviscid Fluids

Mass is neither created nor destroyed, which leads to the *conservation of mass*:

$$\frac{\mathrm{d}}{\mathrm{d}t} \int_{\sigma(t)} \rho(\mathbf{x}, t) \mathrm{d}\mathbf{x} = 0 \tag{5.2}$$

Using the *material derivative*

$$\frac{\mathrm{d}}{\mathrm{d}t} F(\mathbf{x}, t) := \frac{\partial F}{\partial t}(\mathbf{x}, t) + \mathbf{u}(\mathbf{x}, t) \cdot \nabla F(\mathbf{x}, t), \tag{5.3}$$

which is essentially the time derivative along the trajectory of a particle in the fluid, we arrive at the strong formulation of the *continuity equation*

$$\partial_t \rho + \nabla \cdot (\rho \mathbf{u}) \;=\; 0 \tag{5.4}$$

In incompressible flows there is an extra condition which enforces incompressibility

$$\nabla \mathbf{v} = 0 \tag{5.5}$$

The second conservation law is *momentum conservation*, which guarantees that the rate of change of total momentum of a part of a fluid formed by the same particles at any time instant is equal to the forces acting on this part, this leads to the second law

$$\partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p \,\mathrm{Id}) \;=\; -\rho g \mathbf{k} \tag{5.6}$$

where Id is the identity matrix in $\mathbb{R}^2$ and $\mathbf{k}$ the unit vector in the vertical direction.

And last, there is *energy conservation*, so energy in each part of a fluid is neither destroyed nor created, which leads to the last law governing inviscid fluids

$$\partial_t (\rho \theta) + \nabla \cdot (\rho \theta \mathbf{u}) \;=\; 0 \tag{5.7}$$

For inviscid fluids there is no friction and hence a term involving viscosity does not enter.

5.5 Weak and Strong Solutions

If the original problem was to find a differentiable function u defined on the open set W such that

$$P(x,\partial)u(x) = 0 \text{ for all } x \in W$$

(a so-called *strong solution*), then an integrable function u would be said to be a *weak solution* if

$$\int_W u(x)Q(x,\partial)\varphi(x)\,\mathrm{d}x = 0$$

for every smooth function φ with compact support in W.

5.6 Euler Equations

The above conservation laws combined as a set are called the Euler equations. The Euler equations form a set of hyperbolic conservation laws which are non-linear. They can be used to describe the dynamics of a compressible gas or liquid. Since they neglect the effects of both viscous stress as well as heat flux, they are only accurate in the high temperature limit.

$$\begin{aligned} \partial_t \rho + \nabla\cdot(\rho\mathbf{u}) &= 0 \\ \partial_t(\rho\mathbf{u}) + \nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u} + p\,\mathrm{Id}) &= -\rho g\mathbf{k} \\ \partial_t(\rho\theta) + \nabla\cdot(\rho\theta\mathbf{u}) &= 0\,, \end{aligned} \tag{5.8}$$

This formulation is in *primitive variables* since they are expressed in terms of the density ρ, pressure p and the velocity $\mathbf{v}$. The potential temperature θ can be obtained from the equation of adiabatic process in an ideal gas:

$$\theta = T(p)\left(\frac{p_0}{p}\right)^{R/c_p}.$$

where T is the temperature of air at pressure p.

The logarithm of the potential temperature is proportional to the entropy of the system. The potential temperature can be thought of as the temperature of a volume of fluid which has been brought up adiabatically to a height with a given pressure p_o.

The system of differential equations has more unknowns than conditions, so we need an additional equation for closure of the set of equations. In order to close the system we determine pressure from the equation of state

$$p = p_0 \left(\frac{R\rho\theta}{p_0} \right)^{\gamma},$$

where $p_0 = 10^5$ Pa is the reference pressure.

In order to avoid numerical instabilities due to the multi-scale behavior of (5.8), numerical simulations are typically realized for perturbations from an equilibrium state.

In many geophysical applications, flows can be considered as a perturbation of some reference equilibrium state. For example, atmospheric flows are typically represented as a perturbation over the background hydrostatic state $(\bar{\rho}, \bar{\mathbf{u}}(=0), \bar{p}, \bar{\theta})$ [120, 121],

$$\frac{\partial \bar{p}}{\partial y} = -\bar{\rho} g.$$

Here we assume $\bar{\theta} = 300\text{K}$ and $\bar{\rho} = \frac{p_0}{R\bar{\theta}} \bar{\pi}^{\frac{c_v}{R}}$, $\bar{p} \equiv p(\bar{\rho}, \bar{\theta}) = p_0 (\frac{R\bar{\rho}\bar{\theta}}{p_0})^{\gamma}$ with the Exner pressure $\bar{\pi}(x, y) := 1 - gy/(c_p \bar{\theta})$.

The perturbations in our case can be expressed as

$$\rho' = \rho - \bar{\rho},\ \theta' = \theta - \bar{\theta},\ p' = p - \bar{p}.$$

leading to these modified Euler equations for the perturbations

$$\begin{aligned} \partial_t \rho' + \nabla \cdot (\rho \mathbf{u}) &= 0 \\ \partial_t (\rho \mathbf{u}) + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u} + p' \,\mathrm{Id}) &= -\rho' g \mathbf{k} \\ \partial_t (\rho \theta') + \nabla \cdot (\rho \theta \mathbf{u}) &= 0. \end{aligned} \tag{5.9}$$

We rewrite (5.9) in the form of hyperbolic balance law for the vector variable $\mathbf{q} = (\rho, \rho u, \rho v, \rho\theta)^{\mathrm{T}}$

$$\frac{\partial \mathbf{q}}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}) = \mathbf{S}(\mathbf{q}), \tag{5.10}$$

where

$$\mathbf{F}(\mathbf{q}) = \begin{pmatrix} \rho \mathbf{u} \\ \rho \mathbf{u} \otimes \mathbf{u} + p' \,\mathrm{Id} \\ \rho \theta \mathbf{u} \end{pmatrix}, \qquad \mathbf{S}(\mathbf{q}) = \begin{pmatrix} 0 \\ -\rho' g \mathbf{k} \\ 0 \end{pmatrix}$$

is the nonlinear flux function and the source term, respectively. We should note that in our numerical experiments we will also use a stabilization through the artificial viscosity [121, 122], which results in the following source term

$$\mathbf{S}(\mathbf{q}) = \begin{pmatrix} 0 \\ -\rho' g\mathbf{k} + \nabla \cdot (\mu\rho\nabla\mathbf{u}) \\ \nabla \cdot (\mu\rho\nabla\theta') \end{pmatrix}, \quad \mu > 0 \text{ is an artificial viscosity parameter.}$$

Eq. (5.10) will be approximated in space by the discontinuous Galerkin method and in time by the explicit Runge-Kutta scheme. How this is done is the subject of the next sections.

5.7 Numerical Solutions of Hyperbolic Equations

There are many different approaches to hyperbolic equations numerically, and all of them involve some kind of discretization of space and time. The simplest method is the *Finite Differences* method, where space is discretized into a grid, and the exact solution is computed at each of the grid nodes.

Numerical inaccuracy is introduced mainly for the spacial derivatives, since they are represented as *difference quotients*

$$\frac{\partial \mathbf{F}(\mathbf{x})}{\partial \mathbf{x}} \to \frac{\Delta \mathbf{F}(\mathbf{x})}{\Delta \mathbf{x}} \tag{5.11}$$

between neighboring grid nodes. This difference quotient is supposed to describe the exact solution with as much accuracy as possible. Problems arise when the exact solution of the equations is expected to have a steep slope or even a discontinuity.

A different approach are *Finite Elements* methods, where there is no spacial discretization, but instead the problem is solved as a linear combination $\mathbf{q}_h(\mathbf{x})$ in a solution space spanned by a set of basis functions.

$$\mathbf{q}_h(\mathbf{x}, t) = \sum_i^N \mathbf{q}_i(t)\psi_i(\mathbf{x}) \tag{5.12}$$

Since the single basis functions are not time dependent, and only the coefficients $\mathbf{q}_i(t)$ have a time dependence, the spacial derivatives can be precomputed and are known at any time. This method is still not very suited to cope with discontinuities, since representing such a solution requires a large set of solution space basis functions. Yet another solution strategy, which is able to represent discontinuities way better than both of the previous approaches is the *Finite Volume* method.

This method is — similar to the *Finite Differences* method — based on a discretization of space into cells.

In contrast to *Finite Differences* however, the goal is not to find exact solution at specific points in space, but to average the solution over a specific volume cell. Time evolution is represented via flux functions between adjacent volume cells, which can guarantee exact fulfillment of a conservation law. The calculation of the flux is the exact solution of the Riemann Problem [123, 124], since it consists of a conservation law and a piecewise constant initial value problem having a single discontinuity. This is why Finite Volume fluid dynamics solver are essentially Riemann solvers.

The method used in this chapter is the Discontinuous Galerkin (DG) method, which borrows from both the *Finite Elements* as well as from the *Finite Volume* method. This allows the DG method to achieve a good description of smooth solutions as well as discontinuous solutions such as shock waves.

The problem is still solved in a smooth *Finite Elements* space, but only for each volume cell thus allowing discontinuities between volume cells. Time evolution is still represented via flux functions between the interfaces of the volumes, where the discontinuities are.

An important aspect, very much so in the light of this thesis, is good efficieny of the *Discontinuous Galerkin method* in terms of parallel computation [125, 126]. Communication between parallel threads is basically limited to the flux between grid cells, and in contrast to more sophisticated *Finite Volume* methods, the flux is only calculated from directly adjacent volume cells.

5.8 Discontinuous Galerkin Method

We follow [121, 127, 128] and derive the strong formulation of (5.10). Let us divide the computational domain Ω into a finite number of mesh cells Ω_e with a boundary $\partial\Omega_e$.

In our numerical experiments we work with triangular mesh elements Ω_e and use the nodal basis functions $\{\psi_j, j = 1, \ldots, N\}$, N is a number of degrees of freedom. Now, multiplying (5.10) with a nodal basis $\psi_i(\mathbf{x})$, integrating over Ω_e we start with

$$\int_{\Omega_e} \left(\frac{\partial \mathbf{q}(\mathbf{x},t)}{\partial t} + \nabla \cdot \mathbf{F}(\mathbf{q}(\mathbf{x},t)) - \mathbf{S}(\mathbf{q}(\mathbf{x},t)) \right) \psi(\mathbf{x}) \mathrm{d}\mathbf{x} = 0. \tag{5.13}$$

The divergence needs to be in front of the test function $\psi(\mathbf{x})$, so we apply integration by parts

$$\int_{\Omega_e} \left(\frac{\partial \mathbf{q}(\mathbf{x},t)}{\partial t} + \mathbf{F}(\mathbf{q}(\mathbf{x},t)) \cdot \nabla - \mathbf{S}(\mathbf{q}(\mathbf{x},t)) \right) \psi(\mathbf{x}) \mathrm{d}\mathbf{x} = - \int_{\partial\Omega_e} \mathbf{F}(\mathbf{q}(\mathbf{x},t)) \psi(\mathbf{x}) \mathrm{d}S \tag{5.14}$$

where $\mathrm{d}S$ is a surface element. This equation holds for any of the n_e elements Ω_e of the domain Ω, so we have n_e independent equations. We need boundary conditions to solve them. So when working with a numerical solution which is a linear combination of the basis functions as in equation 5.12, we need to replace the flux $\mathbf{F}$ on the right side with a modified flux $\mathbf{F}^*$.

$$\int_{\Omega_\mathrm{e}} \left(\frac{\partial \mathbf{q}_h(\mathbf{x},t)}{\partial t} + \mathbf{F}(\mathbf{q}_h(\mathbf{x},t)) \cdot \nabla - \mathbf{S}(\mathbf{q}_h(\mathbf{x},t)) \right) \psi(\mathbf{x}) \mathrm{d}\mathbf{x} = - \int_{\partial\Omega_\mathrm{e}} \mathbf{F}^* \psi(\mathbf{x}) \mathrm{d}S \tag{5.15}$$

Now, applying integration by parts again and introducing the special basis ψ_i, we obtain the *strong formulation* [129]

$$\begin{aligned} \int_{\Omega_e} \left(\frac{\partial \mathbf{q}_h}{\partial t} + \nabla \cdot \mathbf{F}_h - \mathbf{S}_h \right) \psi_i(\mathbf{x}) \mathrm{d}\mathbf{x} \quad &= \\ \int_{\partial\Omega_e} \quad & (\mathbf{F}_h - \mathbf{F}_h^*)) \, \psi_i(\mathbf{x}) \mathrm{d}S, \quad i = 1, \ldots, N. \end{aligned} \tag{5.16}$$

with $\psi_i(\mathbf{x})$ the basis functions of the numerical solution $\mathbf{q}_h = \mathbf{q}_h(\mathbf{x},t)$. Here, $\mathbf{F}_h$ and $\mathbf{F}_h^*$ are the numerical solutions of $\mathbf{F}$ and the modified flux $\mathbf{F}^*$, and $\mathbf{S}_h$ is the numerical solution of $\mathbf{S}$.

With a specific basis $\psi_i, i = 1, ..., N$, the time evolution of the coefficients $\mathbf{q}_i$ of the numerical solution $\mathbf{q}_h$ can be expressed as

$$\frac{\partial q_i}{\partial t} = - \int_{\Omega_e} \psi_i(\mathbf{x}) \left(\nabla \cdot \mathbf{F}_h - \mathbf{S}_h \right) \mathrm{d}\mathbf{x} + \int_{\partial\Omega_e} \psi_i(\mathbf{x}) \left(\mathbf{F}_h - \mathbf{F}_h^* \right) \mathrm{d}S \tag{5.17}$$

As in [130] Lagrange polynomials are used for the basis functions ψ_j with the Fekete points for the interpolation and the Gauss points for the integrations. In most simulations using the discontinuous Galerkin method for atmospheric flows one-dimensional numerical flux functions, such as the Rusanov flux

$$\mathbf{F}_N^* = \frac{1}{2} \left[\mathbf{F}(\mathbf{q}_N^L) + \mathbf{F}(\mathbf{q}_N^R) - \lambda \hat{\mathbf{n}} (\mathbf{q}_N^R - \mathbf{q}_N^L) \right] \tag{5.18}$$

have been used in [121, 128, 130] and references therein. In [131], a numerical study of the performance of different numerical fluxes for the discontinuous Galerkin method has been presented.

The novelty of our work relies on the application of a multidimensional evolution operator (section 5.9) in order to compute $\mathbf{F}^*$, and make use of parallel computation on the GPU to effectively hide the additional computational effort.

5.9 Multidimensional EG Operator

What follows is a description of the approximate evolution operator that is based on the theory of bicharacteristics for multidimensional hyperbolic conservation laws. My work did not include the derivation of this operator. A detailed derivation will be presented in a forthcoming paper [119]. We follow the derivation presented in [110]. The starting point is equation (5.9), which we will rewrite in a quasi-linear form using the primitive variables $\mathbf{w} = (\rho', u, v, p')$

$$\partial_t \mathbf{w} + \underline{A}_1(\mathbf{w})\,\partial_x \mathbf{w} + \underline{A}_2(\mathbf{w})\,\partial_y \mathbf{w} = \mathbf{s}(\mathbf{w}) \tag{5.19}$$

with

$$\underline{A}_1 = \begin{pmatrix} u & \rho & 0 & 0 \\ 0 & u & 0 & \frac{1}{\rho} \\ 0 & 0 & u & 0 \\ 0 & \gamma p & 0 & u \end{pmatrix}, \; \underline{A}_2 = \begin{pmatrix} v & 0 & \rho & 0 \\ 0 & v & 0 & 0 \\ 0 & 0 & v & \frac{1}{\rho} \\ 0 & 0 & \gamma p & v \end{pmatrix}, \; \mathbf{s} = -\begin{pmatrix} \partial_y \bar{\rho}\, v \\ 0 \\ \frac{\rho'}{\rho} g \\ \partial_y \bar{p}\, v \end{pmatrix}. \tag{5.20}$$

To obtain the quasi-linear form of equation 5.9, we omit the advection term $\rho \mathbf{u} \otimes \mathbf{u}$ in the second equation. In this case, the quasi-linear form is expressed using the pressure p, but the temperature can be used as well. Using the above thermodynamic relationship for $\bar{\rho}, \bar{p}$ we obtain

$$\partial_y \bar{\rho} = -\frac{p_0\, c_v\, g}{(R\bar{\theta})^2 c_p} \left(1 - \frac{g\bar{y}}{c_p\,\bar{\theta}}\right)^{\frac{c_v}{R}-1}, \quad \partial_y \bar{p} = -\frac{g\, p_0}{R\,\bar{\theta}} \left(1 - \frac{g\bar{y}}{c_p\,\bar{\theta}}\right)^{\frac{c_v}{R}}.$$

We cannot diagonalize $\mathbf{A}_1$ and $\mathbf{A}_2$ at the same time, so the closest we can get is to diagonalize the matrix pencil

$$\underline{P} := \underline{A}_1 n_x + \underline{A}_2 n_y.$$

where $\|(n_x, n_y)\| = 1$, so (n_x, n_y) describes a circle in the $x - y$ plane with radius 1, such as the parametrization

$$n_x = \cos(\phi), \quad n_y = -\sin(\phi), \quad \phi \in [0, 2\pi).$$

Since the equation is hyperbolic the matrix pencil has real eigenvalues and a full set of linearly independent eigenvectors.

To linearize (5.19) we *freeze* the Jacobian matrices $\underline{A}_1, \underline{A}_2$ at a suitable intermediate state $\tilde{\rho}', \tilde{u}, \tilde{v}, \tilde{p}'$.

The eigenvalues of $\underline{P}$ are

$$\begin{aligned} \lambda_1 &= \tilde{u}\, n_x + \tilde{v}\, n_y - a, \\ \lambda_2 &= \lambda_3 = \tilde{u}\, n_x + \tilde{v}\, n_y, \\ \lambda_4 &= \tilde{u}\, n_x + \tilde{v}\, n_y + a, \end{aligned}$$

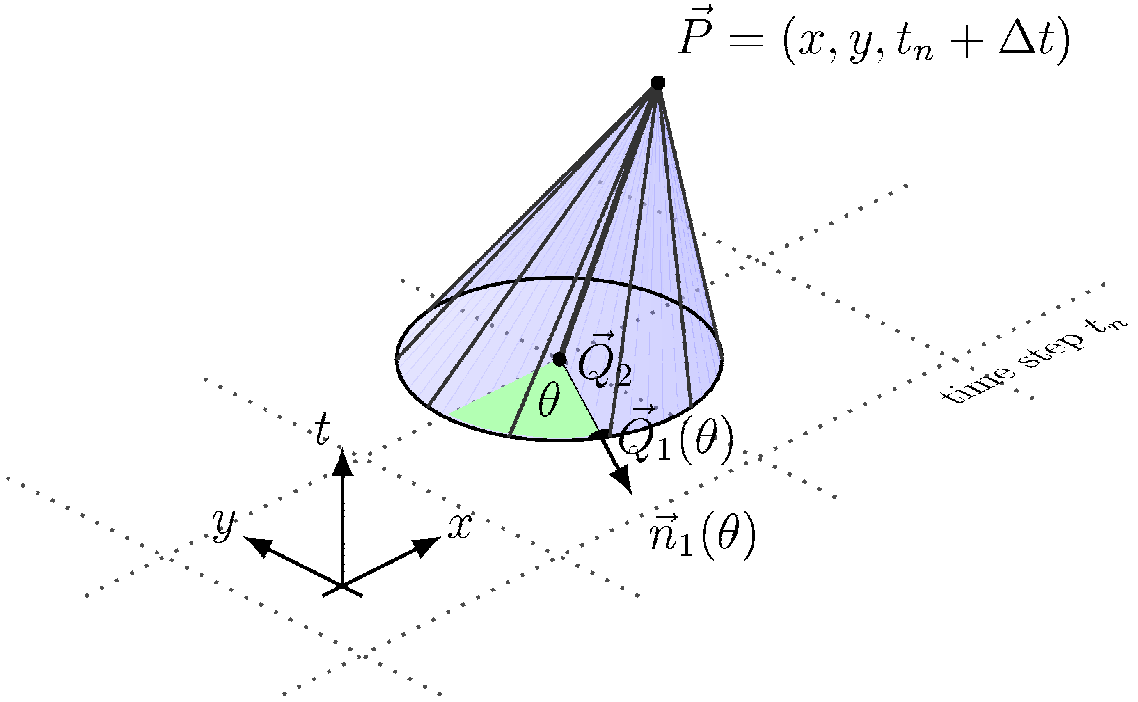


Figure 5.1: Bicharacteristic cone used for the *EG* evolution operator. The figure is the same as in [110].

where $a := \sqrt{\gamma \frac{\tilde{p}}{\tilde{\rho}}} = \sqrt{\gamma R\theta \left(\frac{\rho R\theta}{p0}\right)^{\frac{R}{c_v}}}$ is a sonic speed.

Now we multiply (5.19) by a matrix $\underline{R}^{-1}$, where the matrix $\underline{R}$ consists of the right eigenvectors of $\underline{P}$, so we can rewrite (5.19) using the so-called characteristic variables $\mathbf{v} = \underline{R}^{-1}\mathbf{w}$

$$\partial_t \mathbf{v} + \underline{B}_1 \partial_x \mathbf{v} + \underline{B}_2 \partial_y \mathbf{v} = \mathbf{r},$$

where $\mathbf{r} := \underline{R}^{-1}\mathbf{s}(\mathbf{w})$. Equivalently, we have

$$\partial_t \mathbf{v} + \mathrm{diag}(\underline{B}_1)\partial_x \mathbf{v} + \mathrm{diag}(\underline{B}_2)\partial_y \mathbf{v} = \mathbf{S} + \mathbf{r} \tag{5.21}$$

with

$$\mathbf{S}(\mathbf{x}, \theta) := -\left[(\underline{B}_1 - \mathrm{diag}(\underline{B}_1))\partial_x \mathbf{v} + (\underline{B}_2 - \mathrm{diag}(\underline{B}_2))\partial_y \mathbf{v}\right].$$

Integrating each equation of (5.21) along the corresponding bicharacteristic

$$\frac{d\mathbf{x}_j}{dt} := [\underline{B}_{1,jj}, \underline{B}_{2,jj}]^\top,\ j = 1, \ldots, 4,$$

we obtain after some lengthy manipulations [119] the following exact integral representation

$$\begin{aligned}
\rho'(\mathbf{P}) &= \frac{\tilde{\rho}}{2\pi a}\int_0^{2\pi}\left[-\cos(\theta)\,u(\mathbf{Q}_1(\theta)) - \sin(\theta)\,v(\mathbf{Q}_1(\theta)) + \frac{1}{\tilde{\rho}\,a}p'(\mathbf{Q}_1(\theta))\right]d\theta \\
&+ \rho'(\mathbf{Q}_2) - \frac{p'(\mathbf{Q}_2)}{a^2} \\
&- \frac{\tilde{\rho}}{2\pi a}\int_0^{2\pi}\int_{t_n}^{t_n+\Delta t}\beta(t,\theta)\,dt\,d\theta \\
&- \frac{\tilde{\rho}}{2\pi a}\int_0^{2\pi}\int_{t_n}^{t_n+\Delta t} -\sin(\theta)g\frac{\rho'}{\rho}(\mathbf{x}_1(t,\theta)) + \frac{v(\mathbf{x}_1(t,\theta))}{\tilde{\rho}\,a}\partial_y\bar{p}\,dt\,d\theta \\
&+ \int_{t_n}^{t_n+\Delta t} v(\mathbf{x}_2(t))\left(-\partial_y\bar{\rho} + \frac{\partial_y\bar{p}}{a^2}\right)dt
\end{aligned} \tag{5.22}$$

$$\begin{aligned}
u(\mathbf{P}) &= \frac{1}{2\pi}\int_0^{2\pi}\left[-\frac{p'(\mathbf{Q}_1(\theta))}{\tilde{\rho}a}\cos(\theta) + u(\mathbf{Q}_1(\theta))\cos^2(\theta) + v(\mathbf{Q}_1(\theta))\sin(\theta)\cos(\theta)\right]d\theta \\
&+ \frac{1}{2}u(\mathbf{Q}_2) \\
&+ \frac{1}{2\pi}\int_0^{2\pi}\int_{t_n}^{t_n+\Delta t}\cos(\theta)\,\beta(t,\theta)\,dt\,d\theta \\
&+ \frac{1}{2\pi}\int_0^{2\pi}\int_{t_n}^{t_n+\Delta t} -\sin(\theta)\cos(\theta)g\frac{\rho'}{\rho}(\mathbf{x}_1(t,\theta)) + \cos(\theta)\frac{v(\mathbf{x}_1(t,\theta))}{\tilde{\rho}\,a}\partial_y\bar{p}\,dt\,d\theta \\
&- \frac{1}{2\tilde{\rho}}\int_{t_n}^{t_n+\Delta t}\partial_x p'(\mathbf{x}_1(t))\,dt\,,
\end{aligned} \tag{5.23}$$

$$\begin{aligned}
v(\mathbf{P}) &= \frac{1}{2\pi}\int_0^{2\pi}\left[-\frac{p'(\mathbf{Q}_1)}{\tilde{\rho}a}\sin(\theta) + u(\mathbf{Q}_1)\cos(\theta)\sin(\theta) + v(\mathbf{Q}_1)\sin^2(\theta)\right]d\theta \\
&+ \frac{1}{2}v(\mathbf{Q}_2) \\
&+ \frac{1}{2\pi}\int_0^{2\pi}\int_{t_n}^{t_n+\Delta t}\sin(\theta)\,\beta(t,\theta)\,dt\,d\theta \\
&+ \frac{1}{2\pi}\int_0^{2\pi}\int_{t_n}^{t_n+\Delta t} -\sin^2(\theta)g\frac{\rho'}{\rho}(\mathbf{x}_1(t,\theta)) + \sin(\theta)\frac{v(\mathbf{x}_1(t,\theta))}{\tilde{\rho}\,a}\partial_y\bar{p}\,dt\,d\theta \\
&- \frac{1}{2\tilde{\rho}}\int_{t_n}^{t_n+\Delta t_n}\partial_y p'(\mathbf{x}_2(t))\,dt - \frac{1}{2}g\int_{t_n}^{t_n+\Delta t}\frac{\rho'}{\rho}(\mathbf{x}_2(t))\,dt\,,
\end{aligned} \tag{5.24}$$

$$p'(\mathbf{P}) = \frac{1}{2\pi}\int_0^{2\pi}\left[p'(\mathbf{Q}_1(\theta)) - \tilde{\rho}au(\mathbf{Q}_1(\theta))\cos(\theta) - \tilde{\rho}av(\mathbf{Q}_1(\theta))\sin(\theta)\right]d\theta$$

$$- \tilde{\rho} a \frac{1}{2\pi} \int_0^{2\pi} \int_{t_n}^{t_n+\Delta t} \beta(t,\theta)\, dt\, d\theta$$
$$- \frac{1}{2\pi} \int_0^{2\pi} \int_{t_n}^{t_n+\Delta t} -\sin(\theta)\tilde{\rho} a\, g \frac{\rho'}{\rho}(\mathbf{x}_1(t,\theta)) + v(\mathbf{x}_1(t,\theta))\partial_y \bar{p}\, dt\, d\theta. \tag{5.25}$$

Here $\beta(t,\theta) = a\left[\partial_x u\, \sin^2(\theta) - (\partial_y u + \partial_x v)\sin(\theta)\cos(\theta) + \partial_y v\, \cos^2(\theta)\right]$ and $\mathbf{P} = (x, y, t+\Delta t)$, $\mathbf{Q}_1(\theta) = (x - (\tilde{u} - a\cos(\theta))\Delta t,\, y - (\tilde{v} - a\sin(\theta))\Delta t,\, t_n)$, $\mathbf{Q}_2 = (x - \tilde{u}\Delta t,\, y - \tilde{v}\Delta t,\, t_n)$ are respectively the pick and footpoints of the bicharacteristics that generate the mantle of the bicharacteristic cone, see figure 5.9.

To obtain a time explicit approximate evolution operator the above exact integral representation needs to be approximated. First, time integrals along the mantle of the bicharacteristic cone are approximated using the rectangle rule. Integrals along the base perimeter, that obtain $\beta(t_n,\theta)$ terms, are replaced by means of the integration by parts, cf. Lemma 2.1 [132]. Further we approximate $\frac{\rho'}{\rho}$ with $\frac{\rho'}{\bar{\rho}}$ and substitute the condition for hydrostatic balance $\partial_y \bar{p} = -\bar{\rho} g$. This yields $\partial_y \bar{\rho} = -\pi^{-1} \frac{c_v}{c_p R\bar{\theta}} \bar{\rho} g = -\frac{\bar{\rho} g}{\bar{a}^2}$ used in the approximation for $\rho'(\mathbf{P})$. For more details on the derivation of the approximate evolution operator see [119]. This procedure yields finally the desired cell interface values $\mathbf{q}^* \equiv (\rho'(\mathbf{P}), u(\mathbf{P}), v(\mathbf{P}), p'(\mathbf{P})) = EG\, \mathbf{q}_h$ in (5.16). We should point out that all integrals along the base perimeter (sonic circle), i.e. integrals with respect to θ, are evaluated exactly. We make a transformation of the actual triangle to the reference triangle, where the corresponding integrals along the arcs of sonic circle were precomputed with the help of computer algebra package Mathematica.

5.10 Mesh Adaptivity

Most geophysical flows have a multi-scale character with very localized structural phenomena such as interfaces, turbolences and shock waves. To approximate these local structures efficiently, an adaptive mesh resolution can be used in computer simulations. In our numerical experiments were performed using h-adaptive mesh refinement method, where the spatial resolution is adapted by refining or coarsening the mesh cells. We work with the function library AMATOS of Behrens et al. [133], where h-adaptive mesh refinement is based on the space filling curve approach. Analogously as in [121], in the numerical experiments presented below we use a slightly modified, simple refinement criterion

$$\max_{\mathbf{x}\in\Omega_e} \left[\operatorname{sgn}(\theta'_c)\theta'(\mathbf{x},t)\right] \geq \sigma|\theta'_c| \tag{5.26}$$

for the deviation of the potential temperature from the background state $\theta' = \theta - \bar{\theta}$; $\sigma \ll 1$ is a test dependent parameter (for the numerical experiments in

this work we use $\sigma = 0.1$), and θ'_c is the maximal initial amplitude for the perturbation of the potential temperature (discussed below).

If condition (5.26) holds on some element Ω_e, the element will be recursively refined up to a specified finest mesh resolution. In the rest of the computational domain the mesh is adaptively coarsened, see also [121] for further details. We have used the software package of Müller, Giraldo et al. [121, 128] where the discontinuous Galerkin method (5.16) is implemented. We have generalized the package by including the GPU implementation of the EG operator.

5.11 Numerical Experiments

To verify the accuracy and computational performance of the GPU accelerated code, we carry out two test case simulations. The wording on this section is almost identical to [110]. For our tests we have chosen free convection of a smooth warm air bubble as introduced by Giraldo and Restelli [134] as well as free convection of a large warm bubble with a small cold bubble placed on top of the warm one as introduced by Robert [135].

In the first experiment shown in figure 5.2, the warm bubble is placed at $x_c = 500\text{m}$, $y_c = 350\text{m}$ with the initial temperature perturbation:

$$\theta' = \begin{cases} 0 & \text{for} \quad r > r_c \\ (\theta'_c/2)\left[1 + \cos\left(\pi r / r_c\right)\right] & \text{for} \quad r \leq r_c \end{cases}$$

where $\theta'_c = 0.5^{\text{o}}\text{C}$ is the maximal initial amplitude, the bubble radius $r_c = 250\text{m}$, and r the distance to the center of the bubble (x_c, y_c).

In the Robert experiment, two bubbles are placed at $(x_c, y_c) = (500\text{m}, 300\text{m})$ and $(x_c, y_c) = (560\text{m}, 640\text{m})$, for the warm and the cold bubbles, respectively (figure 5.3). The maximal initial temperature amplitudes are $\theta'_c = 0.5^{\text{o}}\text{C}$ and $\theta'_c = -0.15^{\text{o}}\text{C}$, respectively. The profiles of the initial perturbation for the excess potential temperature are given by a Gaussian distribution

$$\theta' = \begin{cases} \theta'_c & \text{for} \quad r \leq r_c \\ \theta'_c \exp\left[-\left(r - r_c\right)^2 / 50^2\right] & \text{for} \quad r > r_c \end{cases}$$

with a flat core of radius $r_c = 150$ for the warm bubble and $r_c = 0$ for the cold bubble.

In both experiments, due to the differences in the air density of the bubbles and the isothermal environment, the initially resting bubbles develop a vertical motion. In the Giraldo–Restelli test shown in figure 5.2 for both the GPU and CPU simulations, the warm bubble rises and deforms symmetrically due to the shear friction with the surrounding air at the warm/cold air interface, adapting a mushroom-like shape gradually. The results for the GPU simulations in the

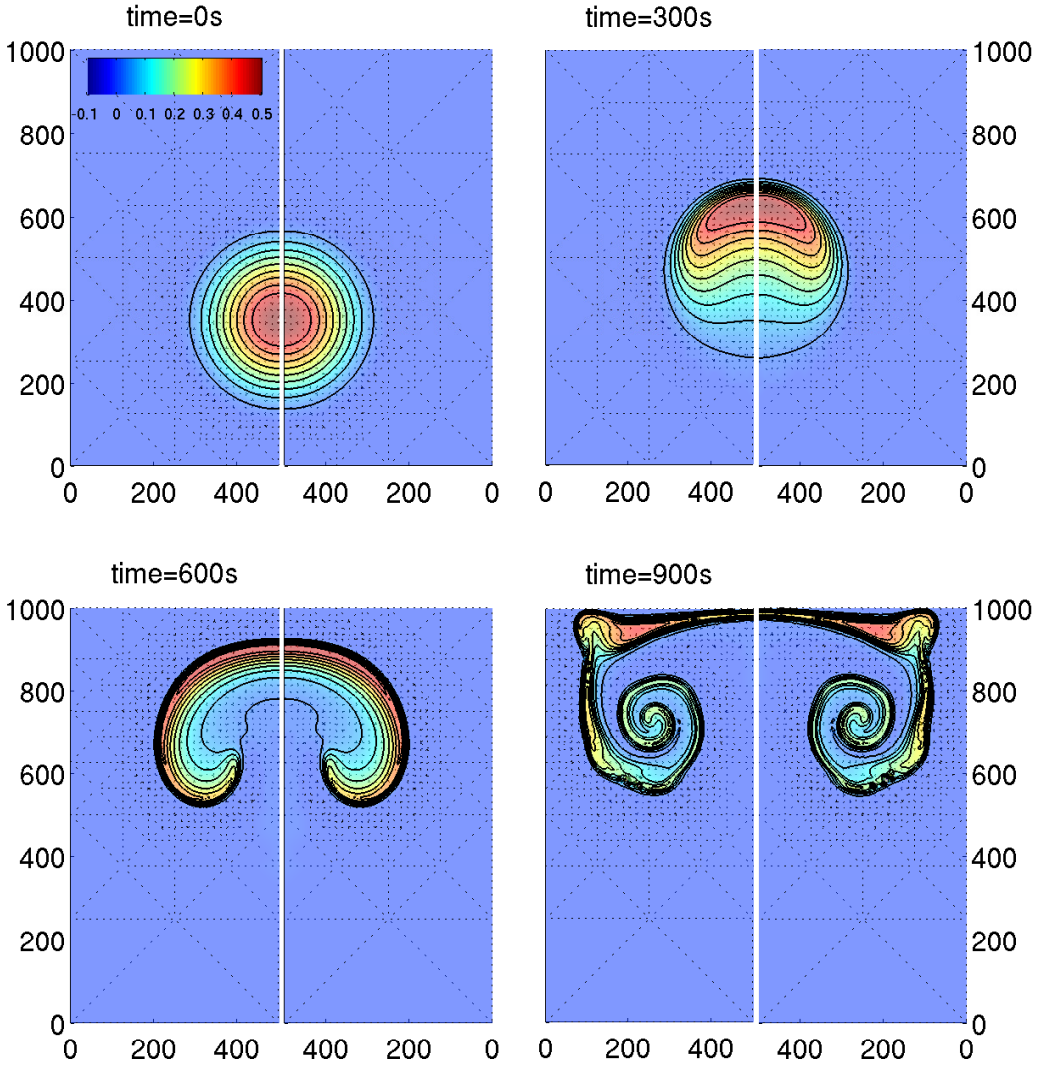


Figure 5.2: Excess potential temperature θ' for the rising thermal bubble experiment as introduced by Giraldo and Restelli [134], on an adaptive resolution grid with the coarse/fine grid resolution levels $n = 1 - 12$, respectively. The left-hand side: CPU simulations, on the right-hand side: accelerated GPU implementation. The real-world domain is 1km$\times$1km (only a half of the squared computational domain is shown in the x-direction); the shortest edge of the adaptive mesh elements corresponds to $\approx$ 11m. The simulation times are as indicated. Contour levels correspond to $\theta' = 0.025$, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, and 0.425°C. The figure was already published in [110].

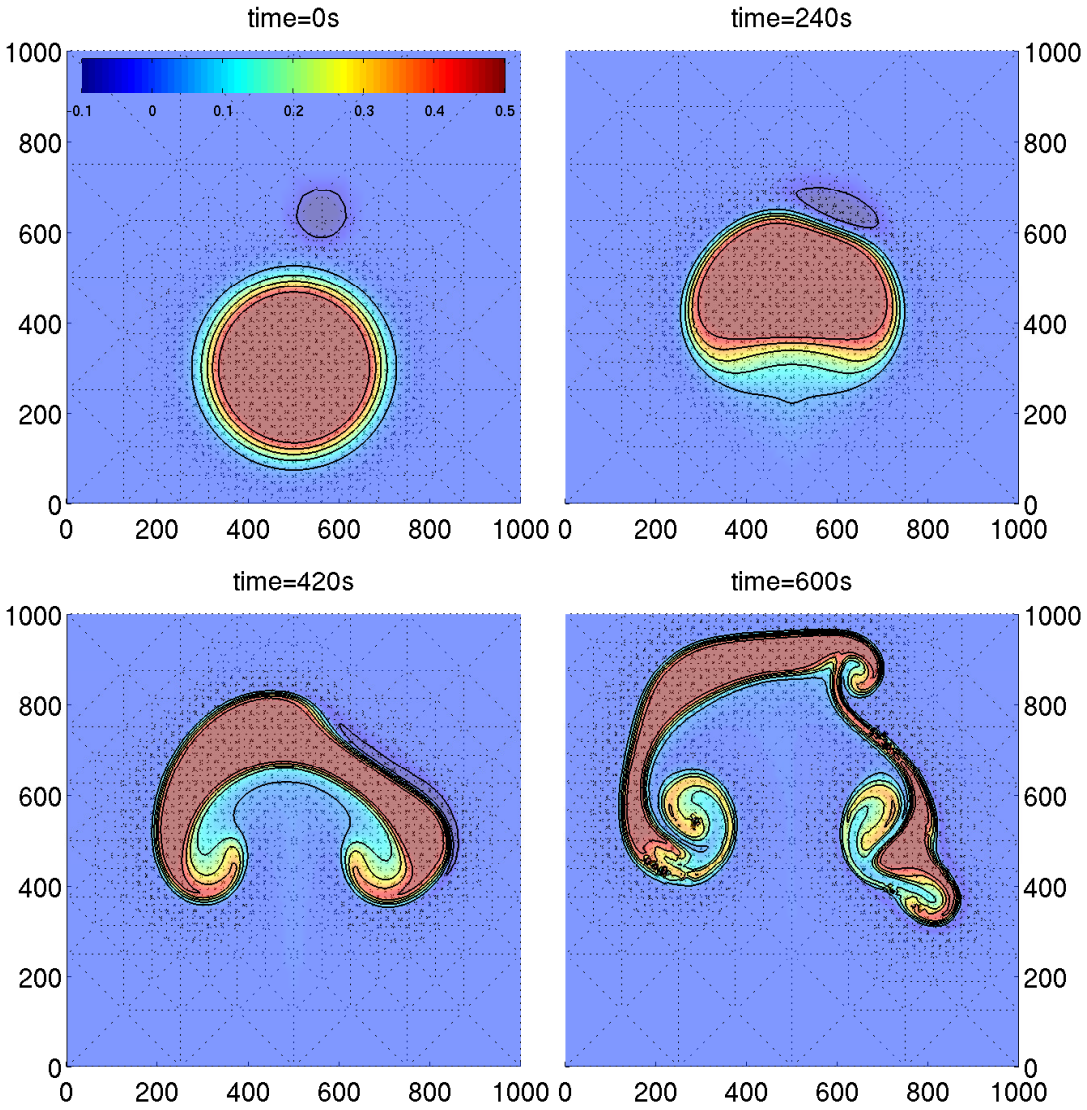


Figure 5.3: Excess potential temperature θ' for a large warm air bubble with a small cold bubble on top, as introduced by Robert [135], obtained by the accelerated GPU simulations on an adaptive resolution grid with the coarse/fine grid resolution levels $n = 2 - 11$, respectively. The real-world domain is 1km×1km; the shortest edge the adaptive grid element corresponds to $\approx$ 15.6m. The simulation times are as indicated. Contour levels correspond to $\theta' = -0.05$, 0.05, 0.15, 0.25, 0.35, and 0.45°C. The figure was already published in [110].

Robert experiment are shown in figure 5.3. The shape of the rising warm bubble is affected in addition by the small cold bubble, which slides downwards along the right-hand side of the interface, destroying the symmetry of the warm bubble.

By comparing the GPU and CPU results, one can recognize no difference between the solutions obtained by CPU and GPU program codes (shown in figure 5.2). This is an important issue since our simulations were performed in single precision on the GPU and in double precision on the CPU. When solving differential equations, slight deviations in initial data or higher inaccuracy of intermediate solution can develop to a different final solution in long time simulations. In order to quantify expected deviations between the GPU and CPU solutions we calculate the L_2 norm

$$L_2(\mathbf{q}) = N^{-1} \left[\sum_{j=1}^{N} \left(\mathbf{q}_{i,\mathrm{CPU}} - \mathbf{q}_{i,\mathrm{GPU}} \right)^2 \right]^{1/2} \tag{5.27}$$

where N is number of degrees of freedom. Of course, the grids in both simulations (performed on GPU and CPU) must have the same structure for this comparison, that cannot be expected in the simulations using adaptive grids. For this reason we have additionally performed simulations on a regular grid with the resolution level $n = 10$, which yields 4096 mesh cells (triangles). This corresponds approximately to the number of mesh elements in simulations on the adaptively refined grid with fine resolution level $n = 12$. In figure 5.4 one can see that discrepancies are indeed present in both the experiments, Giraldo–Restelli and Robert, however, they are of the order of magnitude of rounding errors in the single precision arithmetics and they remain bounded during the simulations. The largest error has been found for the energy variable, $\rho\theta$, which is due to the fact that the potential temperature, θ, is by 2-3 orders of magnitude larger than the other variables in our tests.

5.12 GPU Implementation

As published in [110] we port the most time consuming part of our Computational Fluid Dynamics code, the EG operator, to the GPU while the rest of the code is still being executed on the CPU. The wording on this section is almost identical to the publication. By doing so, we are able to speed up the multidimensional evolution operator by a factor of 30 (GTX580 vs single core Nehalem i7 2.67Ghz), resulting in a roughly sixfold speedup of the overall code.

To evaluate the runtime spent in different parts of the program, we used the Giraldo–Restelli test case with 16384 mesh cells. The total procedure for calculation of EG operator takes up to about 85% of the computation time of the whole

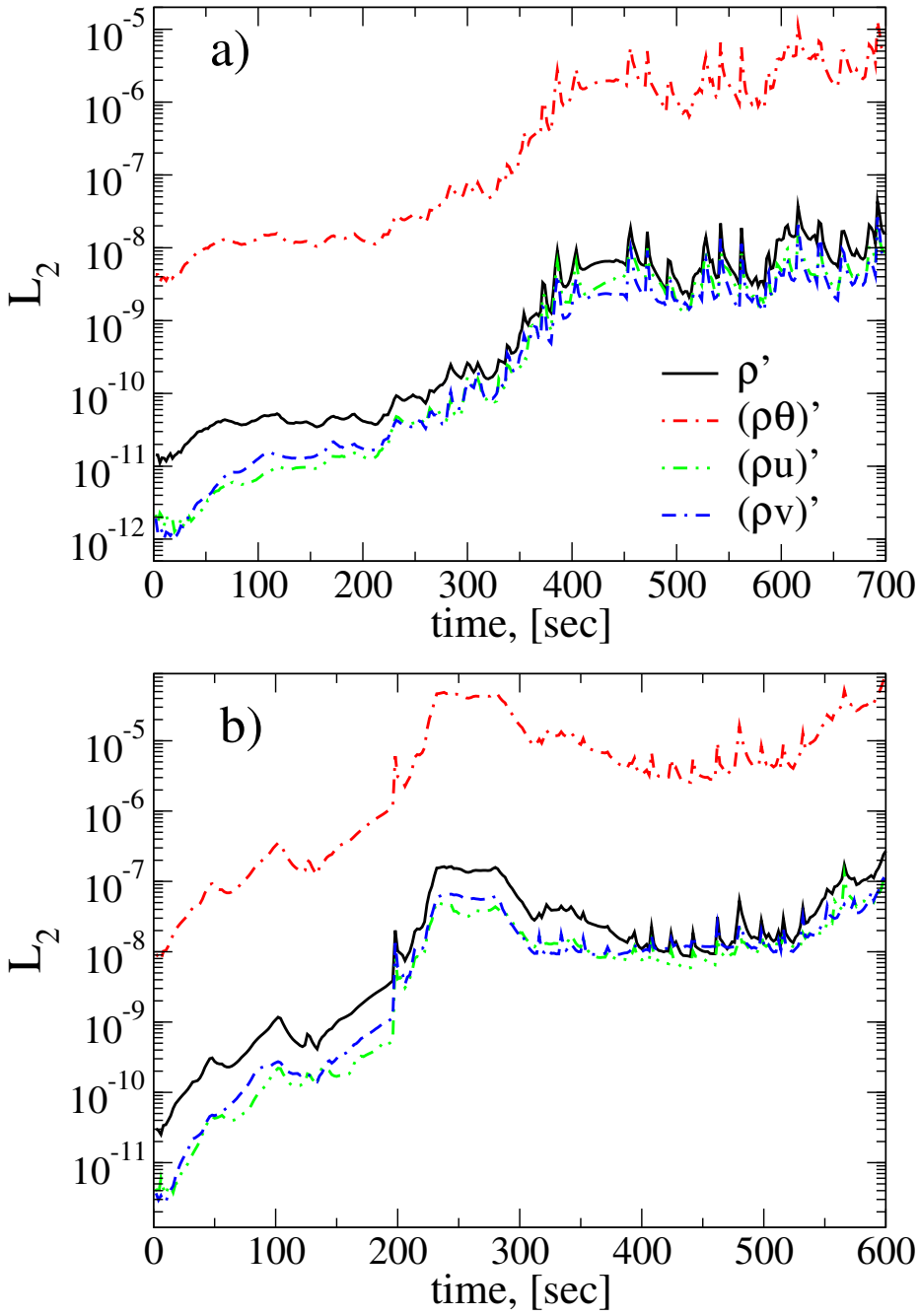


Figure 5.4: The L_2 norm calculated from (eq. 5.27) for the solutions obtained by GPU and CPU codes on regular mesh with $n = 10$ in a) Giraldo–Restelli and b) Robert experiments. The figure was already published in [110].

program in this special case. To date, all of the GPU related work has been done to speed up this procedure, which is structured into a few subprocedures taking the following CPU times in the above described test case.

In the following we list the different parts of the compute_EG procedure and the associated computation time fraction of the complete procedure. The first part of the compute_EG procedure calculates the change of basis (compute quadrature points (0.11%) and compute the basis transformation (3.90%). The second part finds the intersection with the actual wave fronts with each elements edges. The subtasks here are computing the linearized state (1.10%) and compute the wavefront arcs (4.80%). The last part calculates the approximate fields as in eq. (5.22-5.25 ($\rho'(\mathbf{P})$, $u(\mathbf{P})$, $v(\mathbf{P})$, $p'(\mathbf{P})$) which takes by far the largest chunk of computation time (90.01%).

The first step was to port all the code related to the computation of this operator from FORTRAN to C which consisted of several thousand lines of code. Since the data of the main program are stored in the main computer memory, for the calculation of the EG operator on the GPU, the input and output fields have to be transferred to/from the GPU before/after execution. Furthermore, the host program calculates all the fields and quantities in double precision. Therefore, before being transferred to the GPU, the data must be converted into a single precision floating point representation.

We noticed that the data transfer time as well as the conversion time are very low compared to the calculations that are running on the GPU. The last part that takes up over 90% of the execution time was ported to GPU first, to process all the mesh cells in parallel as a heavy weight kernel that needs 63 registers and 1000 bytes of stack memory. This limits GPU occupancy to 33%, but this version of the kernel still performs so fast that the rest of the subprocedures become the new bottleneck of the program. In future work we plan to optimize the kernel to increase the GPU occupancy and, hence, the overall performance of the code.

The next step was to bring the computation of the linearized state, arcs, and basis transformation to the GPU. The computation of basis transformation proved to be the most problematic and least efficient on the GPU, but it was necessary to process it on the GPU since copying the data back and forth between CPU and GPU memory in the middle of the computation is unacceptable.

From equation 3.14, the maximum possible speedup corresponding to the case $t_{\text{GPU,accel}} = 0$ is

$$s_{\max} = \frac{t_{\text{CPU, accel}}}{t_{\text{unaccel}}} + 1. \tag{5.28}$$

In our case the maximum speedup to be expected, if the whole procedure *compute EG* is brought to GPU is $s_{\max} = 6.7$ for grid size 12. This clearly justifies

the effort of a GPU implementation of the procedure in our case.

To benchmark our implementation, we use a NVidia Geforce GTX 580 with a Intel Core i7 Nehalem processor at 2.67 GHz. The measured execution times are reported in Table 1 and compared in figure 5.5 for the CPU and GPU implementations with the grid size level up to $n = 12$, which corresponds to up to 16384 finite volume elements on our computation domain.

Grid size level, n	8	9	10	11	12
Number of finite elements	1024	2048	4096	8192	16384
$t_{\text{CPU, accel}}/\text{sec}$	0.077	0.152	0.307	0.650	1.650
$t_{\text{GPU, accel}}/\text{sec}$	0.0045	0.0065	0.0126	0.0026	0.0569
$s_{\text{EG}} = t_{\text{CPU,accel}}/t_{\text{GPU, accel}}$	17.11	23.38	24.36	25.00	29.70
s_{tot}	5.0	5.4	5.5	5.5	6.6

Table 5.1: Execution times in the Giraldo–Restelli experiment for regular grid of different grid sizes. The net speedup of the GPU implemented parts of the code, s_{EG}, is by a factor up to 30 faster if compared to the CPU execution times. However, due to the non-parallelized parts of the program still running on the CPU, the performance speedup of the whole program, s_{tot}, is much lower (equation 3.14).

Since the kernels are very complex and large, it is hard to predict execution times for finer resolved grids. In our simulations we were restricted to $n = 12$ due to the CFL condition, which relates the sizes of finite elements to the time step used in the explicit time integration scheme for our problems. However, one can see in figure 5.5 that we achieved the most efficient speedup for *compute EG* using our current implementation for $n = 12$. The raw acceleration factor for the GPU implemented *compute EG* procedure is nearly 30 for grid size 12, which means we are nearly reaching the theoretically maximum possible speedup of about 6.7 for this grid size.

Because of the massive speedups gained in the parts of the program executed on the GPU, the execution time of the whole program is largely determined by the parts of the program that remain on the CPU. This means in practice that the differences in speedup for the different grid sizes are barely noticeable in reality and we can expect a relatively stable speedup factor of about 5.0-6.6 for the overall simulation in a real-world example.

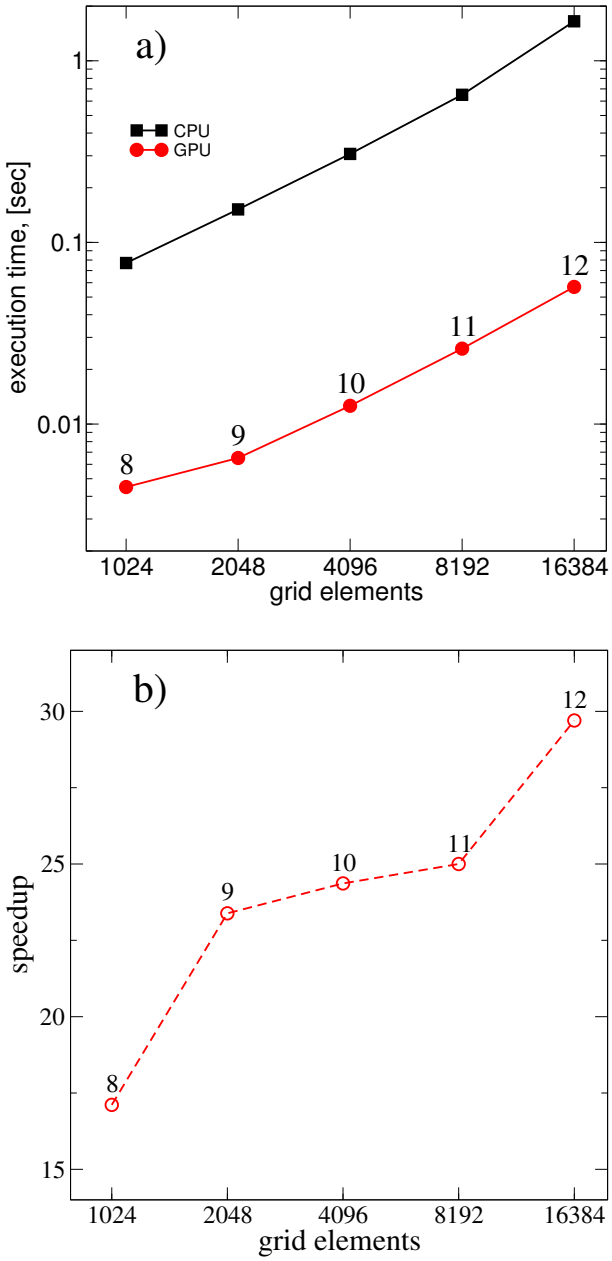


Figure 5.5: a) Execution times in the Giraldo–Restelli experiment for regular grids of different resolutions. b) Numerical speedup of the GPU implemented code for different number of mesh cells in the computation domain. The numbers annotating the symbols are for the grid resolution level, n, (cf. Table 1). The figure was already published in [110].

Conclusion

Our analysis in chapter 4 has brought up a whole range of results concerning the finite temperature roughening transition in the 3D Ising model. This was made possible by only measuring average magnetizations of the lattice planes next to the hard walls. Using a robust method which does not rely on the assumptions of Young's equation, we were able to extract accurate contact angles $\Theta(H)$ for all surface fields H in the range where partial wetting occurs.

A differential equation describing the behavior of the surface free energy in the presence of anisotropy was derived for our system. Combining the results of this measurement with thermodynamic integration methods, this equation could be used to integrate the anisotropic surface tension over a large range of angles. Obtaining surface free energies by integration over the surface magnetizations requires many separate simulations thus resulting in a large computational effort. As we used a parallel GPU (Graphics Processing Unit) implementation of the Ising model, the computational cost was manageable and the results show very good accuracy. The angle dependence of the surface tension has been calculated over a large temperature range from well below the roughening transition temperature T_R up to the vicinity of the bulk critical temperature T_C and is compared with prior predictions of [17] and [113]. Comparing this method with previous measurements in different geometries and with different methods shows good agreement while providing better accuracy, which was achieved especially through the ability to simulate much larger systems.

The temperature dependence of the surface stiffness κ above the roughening temperature T_R was measured by extraction from the curvature of the surface free energy near $\Theta = 90^\circ$. This measurement is comparable to the simulation data obtained by [24] and is in fact in better agreement with theoretical predictions regarding the scaling behavior of the surface stiffness κ, which has been analyzed in detail by [25]. We have developed a low temperature model to explain the small angle behavior of the system below T_R, where the angle stays virtually zero until a critical field H_C, which only has a small system size dependency. When this critical field is overcome, the angle increases rapidly. The critical field H_C has been linked to the Step Free Energy. This made it possible to extract the Step Free Energy from the angle data to analyze the critical behavior of this quantity, which is believed to be of Kosterlitz-Thouless type, thus

featuring a non-polynomial vanishing near T_R. Lastly, by combining different system sizes and comparing their free energy, a method was developed to extract the line tension, also in dependence of the contact angle. The temperature dependence has been investigated by performing this analysis at different temperatures. Our method works well over the whole temperature range as well as for different system sizes.

To gather enough statistics in large scale simulations, we needed a lot of computational resources. This need was met in chapter 3 by implementing a versatile and platform independent parallel simulation code of the Ising model, which performs well in two and in three dimensions. Platform independence could be achieved by abstract interfaces that hide the API (Application Programming Interface) specific details for each platform. The Ising update kernel can be used unchanged for different APIs. GPUs are currently the most cost effective parallel computation devices, so they have been the main target of the implementation. Fair performance comparison is hard between different architectures and especially comparison with non-parallel CPU implementations have to be taken with a grain of salt. The performance is comparable between OpenCL (Open Compute Library) and CUDA (Compute Unified Device Architecture) on Nvidia graphics cards, but we see major differences between vendors.

The conclusions on the performance assessment also hold for the parallelization of the Evolution Galerkin operator in chapter 5. Using Nvidia's CUDA framework, we have obtained a significant speedup for the operator of a factor up to 30 in comparision to a single CPU core. The advantage of this speedup in a real-world simulation may vary, but it can be said that the cost to compute the actual operator can be almost completely hidden by GPU acceleration.

List of Figures

0.1 Simulating nucleation phenomena in the Ising model. Due to enough thermal fluctuations, a spherical droplet can form on an evenly spaced lattice. 9
0.2 A *rough* interface between a dense and a dilute phase in a cubic simulation domain. The angle between the interface and a lattice plane can be adjusted using external fields. 11

1.1 A spherical droplet according to classical nucleation theory has a surface and a volume contribution to the free energy, resulting in a nucleation barrier ΔF which needs to be overcome for a droplet to form. 16
1.2 A $16 \times 16 \times 16$ surface slice of a system of size $120 \times 120 \times 120$ at different temperatures $T = 2.0$ (upper left), $T = 2.5$ (upper right), $T = 3.0$ (lower left) and $T = 3.5$ (lower right). Up spins are depicted as solid blocks while down spins are empty space, in analogy with a lattice gas model. The upper half of the cube therefore depicts a dilute phase (V), while the lower half depicts a dense phase (L). Because of the exchange symmetry there are as many holes in the lower half of the box (which cannot be seen in this visualization) as there are solid boxes in the upper part. The surface exhibits increasing roughness with higher T. Even at low temperatures there are already overhangs and islands in the surface, raising questions about the accuracy of the SOS approximation. 18
1.3 The surface of a tilted interface in the rough phase ($T = 2.6$). Due to enough thermal fluctuations, the shape of the surface is not governed by the underlying lattice structure. Below the roughening transition, the surface takes on a completely different structure (see figure 1.4). The picture is taken from a simulation of a system of size $184 \times 504 \times 504$. 19

1.4 The surface of a tilted interface well below the roughening temperature ($T = 1.5$) has a fundamentally different structure of plateaus and kinks, where each kink can be attributed to a *Step Free Energy*. This example shows the surface of a system of size $184 \times 504 \times 504$ at a wall field of $H = 0.8$, which results in a contact angle of nearly $45°$. 21
1.5 The same system as in figure 1.4, but with a smaller wall field $H = 0.645$. With decreasing wall field, the distance between successive kinks becomes larger, until there is only one kink left. For a single kink to form, the *Step Free Energy* needs to be overcome. At even lower fields, the kink-plateau structure disappears completely and the interface becomes completely flat. 21
1.6 Below the roughening temperature T_R, a *Step Free Energy* appears, with a non-exponential temperature scaling behavior. 22
1.7 Droplet shapes in the 3D Ising model: Contours of constant density $\rho(x, y) = \rho_i$. At temperatures well above the roughening temperature ($T = 4.3$), droplets in the Ising model have a spherical shape (left). Well below the roughening temperature however ($T = 1.0$), they exhibit large anisotropy, thus forming a more and more boxlike shape (right). The images were taken from Schmitz et. al. [8]. 23
1.8 Surface tension anisotropies lead to all kinds of droplet shapes in nature, such as this ice crystal. The crystal shape has a large impact on the dynamics of a nucleation process. The image was taken from the outstanding collection in [20]. 23
1.9 The surface stiffness coefficient of a 3D Ising lattice. When approaching the roughening temperature T_R from above, the surface stiffness rises up to $\frac{\pi}{2}$ at T_R, according to a non-exponential scaling law. Below T_R, it instantly jumps to infinity. When approaching the bulk critical temperature T_C, it has the same universal scaling behavior with the exponent $\mu = (d-1)\nu$ as the surface tension γ, since the γ'' contribution vanishes eventually. 26

1.10 Heterogeneous nucleation can exist in different forms. A situation like the leftmost is the most general situation, with a curved droplet attached to a curved surface. This occurs in nature such as in nucleation on nucleation agents like SO_4 in clouds. The middle configuration has been investigated by [27] and the rightmost is a mostly synthetic situation that occurs in our system due to the choice of boundary conditions, with a flat interface in contact with a hard wall. The contact angle between the surface and the substrate is related to the surface tensions of the different phase boundaries by Young's equation. 27
1.11 A two-phase interface with an angle Θ in contact with a wall. The angle Θ is determined via Young's equation via the surface tensions γ_{LV}, γ_{LW} and γ_{VW}. 28
1.12 The *Wulff construction* is a powerful tool to construct the droplet shape even for a nucleus with surface anisotropy (left side). The *Wulff construction* is a polar plot of the surface tension $\gamma_{VL}(\varphi)$ in dependence of the orientation angle φ with respect to a lattice plane. A horizontal line is drawn that is shifted by the difference in surface tensions between the wall and the liquid and the wall and the vapor above the line of symmetry. The part of the droplet that is above the horizontal line is the droplet shape in contact with the wall. This procedure also works with an isotropic surface tension like the liquid droplet on the right, restoring the prediction of Young's equation 1.9. 28
1.13 The case of a wall attached flat interface. From left to right, a *complete wetting* situation, two different partial wetting configurations and a configuration in the *complete drying* regime. 29
1.14 A two phase interface attached to a wall always forms a one dimensional three-phase boundary. The left figure shows a wall attached droplet, while the right figure shows a flat interface such as in our system. 30
1.15 The layers of the earths atmosphere, from the boundary layer up to the thermosphere. Nucleation phenomena play an important role in several different places. 32
1.16 At different temperatures, different modes of nucleation are possible for ice crystals from water droplets and directly from the surrounding vapor, as well as heterogeneously from aerosol particles. The image was taken from [54]. 34

2.1 Periodic Boundary Conditions (a) can stabilize a *slab* geometry with two flat interfaces. This is only possible if the number of up/down spins is preserved. In a grand canonical configuration, the system will demix into one of the two phases. A canonical Ising simulation scheme is presented in chapter 3, though it is not used in this thesis. With antiperiodic boundary conditions (b), a single flat interface can be stabilized even in the *grandcanonical* ensemble. This is beneficial for our problem, since a *grandcanonical* simulation of the Ising system is a lot easier to parallelize and to run on a GPU. With a nonzero wall field, the mirror symmetry by reflecting the x-axis is broken, so we have to introduce another kind of boundary condition (c) in the y-direction to reflect the new symmetry properties of the system. 39

2.2 The profiles of the average magnetization in the uppermost layer $\langle m_o \rangle(y)$ as a profile along the y-axis reveal a problem that needs to be addressed. At $H = 0$ (black line) the profile is perfectly antisymmetric ($m^o_- = -m^o_+$), so antiperiodic boundary conditions (section 2.2.3) work, since the magnetizations in the bulks are nearly constant even near the system border (which is an important property for later measurements). When $H \neq 0$ however, the symmetry is broken, which leads to border artifacts for antiperiodic boundary conditions (red dotted line). The GAPBC boundary conditions introduced in section 2.2.4 solve this problem. 41

2.3 A full configuration of size $184 \times 504 \times 504$ at $T = 2.0$. This shows the scale of the simulations performed in this thesis. 44

2.4 The checkerboard spin update procedure. Spins are selected and updated in two steps. In the first step, only spins from the white sites of a checkerboard are updated, while in the second step, the remaining spins are updated. After the two steps, each of the spins in the lattice has been updated once. 44

2.5 The density profile in the $x - y$ plane at different Monte Carlo times t. In this specific case, notice the drift of the surface in the middle of the system to the right side. This is not a systematic drift, but the surface performs a random walk. Nevertheless, it is expected to eventually hit one of the borders of the system. 45

2.6 The average of the magnetizations in the uppermost (m_o) and the lowermost (m_u) layer can be easily extracted from a GPU simulation. This data has been extracted from a 10^5 full lattice update run of a $88 \times 504 \times 504$ system. The data has been averaged over about 20 runs. The result is proportional to the mean square displacement of the interface which is expected to perform a random walk. The end to end mean square displacement of a random walk is expected to be a linear function of the amount of steps. A linear dependence can be verified in our data (red line). 46

3.1 Hierarchical parallel execution of *kernels* on an Nvidia GPU. *Kernels* are executed by parallel threads, which are organized in blocks, where the whole block has access to a small shared memory space which can be used to accelerate block local computations. The blocks are arranged in a grid. Each thread in the whole grid can access a slower kind of memory, the global memory. There is also a memory that is local to each thread, and can not be shared between threads at all. Synchronization between threads is possible on the block level, but not on a global level. 49

3.2 At each time step a spin is flipped from up to down or from down to up in dependence on the local interaction energy difference before and after the flip. There are 5 possible energy differences in the two dimensional case. Negative energy differences and energy differences of value 0 are always accepted, so only two exponential factors need to be calculated. This can be done in advance to save computation time. The image was taken from [74]. 52

3.3 Checkerboard layout for parallel thread updates. The image was taken from [74]. 54

3.4 Procedure how each thread processes a 4x4 meta-spin. First, the meta-spin and its neighbors are looked up in global memory. After that, the actual spins are extracted into shared memory and an update pattern is created by evaluating the Metropolis criterion for each of the spins. After that, the new spins are obtained using the update pattern and written back to global memory. The figure was taken from [76]. 60

3.5 Organizing the spin lattice in 128 bit spin blocks makes memory lookup very efficient and the update logic fast. The figure was taken from [73]. 61

3.6 Organizing the spin lattice in 128 bit spin blocks makes memory lookup very efficient and the update logic fast. The figure was taken from [73]. 62
3.7 Extracting and processing a row of spins at a time. The figure was taken from [76]. 63
3.8 (a) shows how periodic boundary conditions can be implemented by simulating 1 excess row of spin blocks on both sides for each dimension. A memory exchange transfer is needed between the red and blue regions to propagate physical information across the border. This scheme can be used to spread the simulation lattice over more than one GPU for each dimension (b). Spin data has to be transferred between the domains as indicated by the red and blue arrows. The memory transfer is necessary after n_b time steps, where n_b is the linear dimension of the spin block, to make sure physical information is properly propagated over the domain borders. (Information spreads by one spin block per time-unit). The figure was taken from [76]. 65
3.9 The time autocorrelation function $g(t)$ in a 2D Ising model of size 496×496. The pattern in which spins are selected for update have a significant effect on the spin correlations between different times. An interesting observation is that a deterministic ABAB checkerboard update leads to the fastest disappearance of correlations in the system. 67
3.10 Single spin flips per nanosecond for a two dimensional system. Performance of OpenCL and CUDA on the Nvidia card are more or less comparable, while AMD clearly lacks behind. Also the largest system size fails to start for an unknown reason. This might be a due to an architectural limitation on AMD cards. Intels OpenCL implementation scales very well and impressively with the amount of cores on the i7 Nehalem processor. The figure was taken from [76]. 69
3.11 Single spin flips per nanosecond for a three dimensional system. The Nvidia card fares again nearly equally well on CUDA and OpenCL. AMDs performance is not much behind in this case, making the use of an AMD card a true alternative. The figure was taken from [76]. 70

4.1 The simulation geometry. Over the boundaries of the y-dimension, (generalized) anti-periodic boundary conditions are transmitted. The boundary conditions in y-direction stabilize a flat interface between the two phases. In x-direction, the simulation domain is bounded by hard walls, which can be adjusted to a external wall field of magnitude $|H|$, so the uppermost and lowermost layers of spins feel the influence of the wall field. When an external H-field is applied to the Ising-spins next to the wall, the interface is tilted with an angle Θ with respect to the wall surface. If the wall field is adjusted so the upper field has a value of $+H$, while the lowermost has a value of $-H$, the symmetry does not favor any of the two phases and the interface gets tilted without a systematic drift in any direction. 76
4.2 The integration path in the T-H plane for the thermodynamic integration. First, an integration along the temperature axis is performed, after that comes the integration along the external field H. 79
4.3 The surface tension $\gamma_0(T)$ (a) for different temperatures from the equation 4.6 as calculated by Hasenbusch et. al. [114]. This data has been verified in our simulation and used as a basis for the integration of the angle dependent tension in section 4.5 and the stiffness calculation in section 4.6. The line tension $\tau(T)$ (b) for different temperatures from the equation 4.7 as calculated by Kim et. al. [42]. The line tension data is used in section 4.9 to construct the contact angle dependent line tension $\tau(\Theta, T)$. 82
4.4 A *grandcanonical* simulation of a system ($88 \times 504 \times 504$) with periodic boundary conditions (PBC) does not form an interface. The wall field of $H = 0.8 \neq 0$ creates a visible thin film of up spins in the lower layer of spins (red circle). The magnetization of the upper layer of spins $\langle m_o^{(-)} \rangle$ is then subtracted from the magnetization of the lower layer $\langle m_u^{(-)} \rangle$ to extract the contact angle using Young's equation. 85
4.5 The magnetization difference $\langle m_o^{(-)} \rangle - \langle m_u^{(-)} \rangle$ of the uppermost and the lowermost layer. Thus when integrated over, this difference can be used to extract the free energy difference seen in figure 4.6 85

4.6 Subtracting the integral of the magnetization of the upper layer from the integral of the magnetization of the lower layer leads to the total free energy of the upper layer and the lower layer combined. This free energy is the free energy attributed to the interfaces of the spins at contact with the wall. 86
4.7 Using equation 4.10, the contact angle can be extracted from the total surface free energy of the system, under the assumption of an isotropic surface tension. The measured data is in full agreement with Winter et. al. [27] for the temperatures that are available for comparison. Simulations are extended to various temperatures down to a temperature of $T = 1.5$ 87
4.8 The geometry and definitions of a two domain state system with antiperiodic boundary conditions in y-direction and a tilted interface between them. The lengths of the spin domains in y-dimension in the uppermost and lowermost layer are denoted by x^o_- and x^u_- for the spin-down majority phase, and x^o_+ and x^u_+ in the spin-up majority phase. 88
4.9 The contact angle for $T = 3.0$ at a system size of $84 \times 504 \times 504$, calculated by the geometric construction in equation 4.17 (in black) is systematically larger than the angle calculated by means of Young's equation (in red). This is a first hint that Young's equation does clearly not hold under these circumstances. We will see that this difference is very valuable to obtain the angle dependence of the surface tension $\gamma(\Theta)$ as will be shown in section 4.5. 90
4.10 The contact angle calculated by equation 4.17 for dhifferent temperatures ranging from $T = 2.5$ up to $T = 3.8$. As revealed by figure 4.9, the curves are systematically larger than prediced by Young's equation (equation 4.10). 90
4.11 Anisotropy values for different temperatures. This data is visualized and compared to literature in figures 4.14 and 4.15. 93
4.12 $\gamma(\Theta)$ for different temperatures in the range between $T = 2.5$ and $T = 3.8$. This data is obtained via numeric integration according to equation 4.19. The data in figure 4.3 was used for $\gamma(90°)$. As can be seen, the variation in the absolute value of γ is rather small, especially above the roughening temperature T_R. When approaching T_R, the variation increases significantly. 94

4.13 Surface anisotropy $\gamma(\Theta)/\gamma(90^\circ)$ for different systems approaching the roughening transition. The data in figure 4.3 was used for $\gamma(90^\circ)$. Lower temperatures feature a significant anisotropy in surface tension which is worth investigating in detail, since it is expected to have an importance in the prediction of nucleation rates and crystal shapes. 95
4.14 Surface anisotropy $\gamma(45^\circ)/\gamma(90^\circ)$ for different temperatures. The data can be compared to [113] and is found to be in good agreement. We are able to predict the value of the anisotropic surface tension for a variety of angles down to the roughening transition temperature T_R. The line only serves as a guide to the eye. 96
4.15 Surface anisotropy $\gamma(45^\circ)/\gamma(90^\circ)$ for different temperatures. We also compared our results to the data of Mon et. al. [17]. The publication indicates a significant error in their simulation data, so our results can provide much more accuracy even below T_R. This makes qualitative investigation of the roughening transition feasible and interesting. The quantities that can be analyzed here are the surface stiffness and the *Step Free Energy* as the order parameter of the roughening transition. At lower temperatures ($T < 2.0$) we do not have any data for contact angles of 45°, since the angle is not reached with our simulation range of $H \in [0, 0.8]$. The line only serves as a guide to the eye. 97
4.16 The constant c in equation 4.20 can be retrieved from the difference of angles $\Theta - \Theta_0$, which have already been measured. It helps evaluating in which range of angles a linear dependence is justified. This data is for a temperature of $T = 3.0$. The red bar indicates the average value over the interval $[90, 85]$, which can be used as an estimate for c. The data is very noisy and not optimal for the retrieval of a value for c. 99
4.17 The constant c in equation 4.20 for three different temperatures. The range of linear dependence gets more narrow with lower temperatures. 100
4.18 The constant c for different temperatures. Unfortunately the method does not produce very satisfying results, especially in the vicinity of the roughening transition. 100
4.19 Surface tension in the vicinity of $\Theta = 90^\circ$ for different temperatures near the roughening transition at T_R. The curve is fitted to equation 4.6.1 to extract the surface stiffness. The fit needs to be restricted to a small angle around 90°, since the expansion in terms of surface stiffness is only accurate up to leading order. . . . 102

4.20 Surface Stiffness $\kappa(90°)/\gamma(90°)$ from our analysis in a system of size $88 \times 504 \times 504$ compared to the results obtained by [24]. Our data reflects the theoretical prediction (see figure 1.6), which predicts a vanishing of the surface stiffness at high temperatures and a divergence of the stiffness at temperatures below T_R very well. The line only serves as a guide to the eye. 103
4.21 Below the roughening transition, the difference between Θ and Θ_0 which has been determined via Young's equation is much less subtle than above the transition. The simulation data is for $T = 2.0$ at a system size of $184 \times 504 \times 504$. The black curve shows the angle calculated from the difference in wall free energies using Young's Equation, while the red line shows the angle Θ calculated from the geometric construction. Below the roughening transition, the system has to reach a critical field H_C before it can form a tilted interface. This critical field is linked to a *Step Free Energy* as explained by the model developed in section 4.7.1. 104
4.22 The three contributions to the low temperature model of a tilted interface with a single kink illustrated in one picture. 105
4.23 For larger inclination angles and larger external fields, the model needs to incorporate multiple step configurations as well. 107
4.24 An illustration of the angle expectation value $\langle\Phi\rangle$ for $f_s = 0.5$ and $T = 2.0$. Taking the occupation of higher order states into account, the saturation angle is much higher, but the behavior in the onset of population in the non-zero angle states does not change. The transition is very sharp and is accurate only in the very low temperature regime. 109
4.25 The contact angle data at $T = 2.0$ for different system sizes L_x in comparison with the theory curve for T=2.0 and a *Step Free Energy* of $f_s = 0.52$. The behavior up to the critical field H_C indicates agreement with our theory in the limit of an infinite system. The large field behavior however is not captured (figure 4.24), which is believed to be governed by more complex interactions such as effective kink-kink repulsions which modify the large field behavior. Nevertheless, the crossover position can be used to estimate the *Step Free Energy* and compare it to literature [17]. 110
4.26 At low temperatures, the contact angle curve resembles more and more to that of the low temperature model since the step at H_C gets sharper with lower temperatures. 111

4.27 The *Step Free Energy* in dependence of the temperature T below the roughening transition T_R shows good agreement with the theoretically predicted non-universal scaling behavior. The error bars approximate the width of the step at H_C. The data points at $T = 1.5$ and $T = 1.8$ are excluded from the least squares fit. 113
4.28 The second derivative $\gamma''(T)$ in dependence on the number of data points included in the fit (section 4.6.1). For the thermodynamic integration, the distance between two data points is $\Delta H = 0.005$. The connecting lines between the data points only serve as a guide to the eye. 114
4.29 The simulation data in a system of size $184 \times 504 \times 504$ in comparison with the exact theoretical prediction $\beta\kappa(T) = \frac{\pi}{2}\left(1 - c \cdot \sqrt{\frac{T-T_R}{T_C}}\right)$. For the blue line, c is chosen according to the analysis of [17, 25], while the red line shows the same scaling law with c choosen according to the outcome of our own analysis of the critical scaling of the *Step Free Energy*. 115
4.30 The free energy differences $\Delta F_{s,\mathrm{int}}(L_z)$ are expected to have only a linear dependence on the linear dimension L_z, if all system sizes are varied so that the products $L_x L_z$ and $L_y L_z$ are kept constant. All graphs are shown for a system of temperature $T = 3.0$. The uppermost graph shows the dependence for $H = 0.15$, the middle one for $H = 0.35$ and the lowermost for $H = 0.45$. 120
4.31 Anisotropic line tension for different temperatures ranging from $T = 2.5$ up to $T = 4.0$. The most important observation is that the line tension decreases for interface orientations that are not parallel to the lattice plane. 121
4.32 Line Tension temperature dependence for different angles $\Theta = 75°$, $\Theta = 60°$ and $\Theta = 45°$. The maximal anisotropy $\frac{\tau(\Theta)}{\tau(90°)}$ is around $T = 3.5$. This is also the temperature of the lowest absolute line tension $\tau(90°)$. The lines between the data points only serve as a guide to the eye. 122

5.1 Bicharacteristic cone used for the EG evolution operator. The figure is the same as in [110]. 132

5.2 Excess potential temperature θ' for the rising thermal bubble experiment as introduced by Giraldo and Restelli [134], on an adaptive resolution grid with the coarse/fine grid resolution levels $n = 1 - 12$, respectively. The left-hand side: CPU simulations, on the right-hand side: accelerated GPU implementation. The real-world domain is 1km$\times$1km (only a half of the squared computational domain is shown in the x-direction); the shortest edge of the adaptive mesh elements corresponds to $\approx$ 11m. The simulation times are as indicated. Contour levels correspond to $\theta' = 0.025$, 0.075, 0.125, 0.175, 0.225, 0.275, 0.325, 0.375, and 0.425°C. The figure was already published in [110]. 136

5.3 Excess potential temperature θ' for a large warm air bubble with a small cold bubble on top, as introduced by Robert [135], obtained by the accelerated GPU simulations on an adaptive resolution grid with the coarse/fine grid resolution levels $n = 2 - 11$, respectively. The real-world domain is 1km$\times$1km; the shortest edge the adaptive grid element corresponds to $\approx$ 15.6m. The simulation times are as indicated. Contour levels correspond to $\theta' = -0.05$, 0.05, 0.15, 0.25, 0.35, and 0.45°C. The figure was already published in [110]. 137

5.4 The L_2 norm calculated from (eq. 5.27) for the solutions obtained by GPU and CPU codes on regular mesh with $n = 10$ in a) Giraldo–Restelli and b) Robert experiments. The figure was already published in [110]. 139

5.5 a) Execution times in the Giraldo–Restelli experiment for regular grids of different resolutions. b) Numerical speedup of the GPU implemented code for different number of mesh cells in the computation domain. The numbers annotating the symbols are for the grid resolution level, n, (cf. Table 1). The figure was already published in [110]. 142

Bibliography

[1] L. Onsager. Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition. *Phys. Rev.*, 65(3-4):117–149, Feb 1944.

[2] M. E. Fisher. Theory of condensation and critical point. *Physics - New York.*, 3, 1967.

[3] H.-O. Heuer. Critical crossover phenomena in disordered Ising systems. *J. Phys. A: Math. Gen.*, 26:L333–L339, 1993.

[4] M. Hasenbusch. Monte carlo studies of the three-dimensional Ising model in equilibrium. *International Journal of Modern Physics C*, 12(07):911–1009, 2001.

[5] M. Campostrini, M. Hasenbusch, A. Pelissetto, P. Rossi, and E. Vicari. Critical exponents and equation of state of the three-dimensional Heisenberg universality class. *Phys. Rev. B*, 65(14):144520, 2002.

[6] H. van Beijeren and I. Nolden. The Roughening Transition. *Topics in Current Physics*, August 1987.

[7] W. Kossel. *Nachr. Ges. Wiss. Goettingen, Mathemat./Physikal. Klasse*, page 135, 1927.

[8] F. Schmitz, P. Virnau, and K. Binder. Monte Carlo tests of nucleation concepts in the lattice gas model. *Phys. Rev. E*, 87(5), MAY 3 2013.

[9] G. Wulff. Zur Frage der Geschwindigkeit des Wachstums und der Auflösung der Kristallflächen. *Z. kristallogr*, 34:449–530, 1901.

[10] T. Bodineau, D. Ioffe, and Y. Velenik. Rigorous probabilistic analysis of equilibrium crystal shapes. *Journal of Mathematical Physics*, 41:1033, 2000.

[11] J. Coninck and F. Dunlop. Partial to complete wetting: A microscopic derivation of the Young relation. *J. Stat. Phys.*, 47(5-6):827–849, 1987.

[12] E. Burkner and D. Stauffer. Monte-Carlo Study of Surface Roughening in the 3-dimensional Ising-model. *Zeitschrift fur Physik B-Condensed Matter*, 53(3):241–243, 1983.

[13] K. K. Mon, D. P. Landau, and D. Stauffer. Interface roughening in the three-dimensional Ising model. *Phys. Rev. B*, 42:545–547, Jul 1990.

[14] M. Hasenbusch, S. Meyer, and M. Pütz. The roughening transition of the three-dimensional Ising interface: A monte carlo study. *J. of Stat. Phys.*, 85(3-4):383–401, 1996.

[15] M. Hasenbusch, M. Marcu, and K. Pinn. High precision renormalization group study of the roughening transition. *Physica A: Statistical Mechanics and its Applications*, 208(1):124–161, 1994.

[16] J. M. Kosterlitz and D. J. Thouless. Ordering, metastability and phase transitions in two-dimensional systems. *Journal of Physics C Solid State Physics*, 6:1181–1203, April 1973.

[17] K.K. Mon, S. Wansleben, DP Landau, and K Binder. Monte Carlo studies of anisotropic surface tension and interfacial roughening in the three-dimensional Ising model. *Physical Review B*, 39(10):7089, 1989.

[18] J.P. van Eerden and H.J.F. Knops. Correlations in the xy model and screw dislocations in the solid-on-solid model. *Phys. Lett. A*, 66(4):334–336, 1978.

[19] C.-N. Yang and T.D. Lee. Statistical theory of equations of state and phase transitions. i. theory of condensation. *Phys. Rev.*, 87(3):404–409, 1952.

[20] Kenneth G. L. Snow flakes and snow crystals. `http://www.its.caltech.edu/~atomic/snowcrystals/`.

[21] K. Binder, M. Bowker, J. E. Inglesfield, and P. J. Rous. *Cohesion and Structure of Surfaces*, volume 4. Elsevier, 1995.

[22] V. Privman. Fluctuating interfaces, surface tension, and capillary waves: an introduction. *International Journal of Modern Physics C*, 3(05):857–877, 1992.

[23] R.E. Rozas and J. Horbach. Capillary wave analysis of rough solid-liquid interfaces in nickel. *Eur. Phys. Let.*, 93(2):26006, 2011.

[24] M. Hasenbusch and K. Pinn. Surface tension, surface stiffness, and surface width of the 3-dimensional Ising model on a cubic lattice. *Physica A: Statistical Mechanics and its Applications*, 192(3):342–374, 1993.

[25] M.E. Fisher and H. Wen. Interfacial Stiffness and the Wetting Parameter - The Simple Cubic Ising-Model. *Phys. Rev. Lett.*, 68(24):3654, JUN 15 1992.

[26] T. Young. An essay on the cohesion of fluids. *Philos. Trans. Roy. Soc. London*, 95:65–87, 1805.

[27] M. Schrader, P. Virnau, D. Winter, T. Zykova-Timan, and K. Binder. Methods to extract interfacial free energies of flat and curved interfaces from computer simulations. *European Physical Journal*, 177:103–127, 2009.

[28] W.L. Winterbottom. Equilibrium shape of a small particle in contact with a foreign substrate. *Acta Metallurgica*, 15(2):303–310, 1967.

[29] R.K.P. Zia and A. Gittis. Effects of gravity on equilibrium crystal shapes: Droplets hung on a wall. *Phys. Rev. B*, 35:5907–5909, 1987.

[30] J. W. Cahn. Critical point wetting. *The Journal of Chemical Physics*, 66:3667, 1977.

[31] D.E. Sullivan. Surface tension and contact angle of a liquid–solid interface. *The Journal of Chemical Physics*, 74:2604, 1981.

[32] C. Ebner and W.F. Saam. New phase-transition phenomena in thin argon films. *Phys. Rev. Lett.*, 38(25):1486, 1977.

[33] R. Pandit, M. Schick, and M. Wortis. Systematics of multilayer adsorption phenomena on attractive substrates. *Phys. Rev. B*, 26(9):5112, 1982.

[34] H. Nakanishi and M.E. Fisher. Multicriticality of wetting, pre-wetting, and surface transitions. *Phys. Rev. Lett.*, 49(21):1565–1568, 1982.

[35] E. Brezin, B.I. Halperin, and S. Leibler. Critical Wetting in 3 Dimensions. *Phys. Rev. Let.*, 50(18):1387–1390, 1983.

[36] P. Bryk and K. Binder. Non-mean-field behavior of critical wetting transition for short-range forces. *Phys. Rev. E*, 88:030401, Sep 2013.

[37] G. Navascues and P. Tarazona. Contact-angle and line tension dependence on curvature. *Chem. Phys. Lett.*, 82(3):586–588, 1981.

[38] J. Y. Wang, S. Betelu, and B. M. Law. Line tension effects near first-order wetting transitions. *Phys. Rev. Lett.*, 83:3677–3680, Nov 1999.

[39] M. Schick, K. Katsov, and M. Muller. The central role of line tension in the fusion of biological membranes. *Molecular Physics*, 103(21-23):3055–3059, NOV-DEC 2005.

[40] L. Schimmele, M. Napiorkowski, and S. Dietrich. Conceptual aspects of line tensions. *J. Chem. Phys.*, 127(16), OCT 28 2007.

[41] D. Winter. Diploma thesis. 2009.

[42] S. Kim. Diploma thesis. 2011.

[43] R.J. Charlson, S.E. Schwartz, J.M. Hales, R.D. Cess, J.A. Coakley, J.E. Hansen, and D.J. Hofmann. Climate Forcing by Anthropogenic Aerosols. *Science*, 255(5043):423–430, JAN 24 1992.

[44] S. Twomey. Atmospheric aerosols. 1977.

[45] D.V. Spracklen, K.S. Carslaw, M. Kulmala, V.-M. Kerminen, G.W. Mann, and S.-L. Sihto. The contribution of boundary layer nucleation events to total particle concentrations on regional and global scales. *Atmospheric Chemistry and Physics Discussions*, 6(4):7323–7368, 2006.

[46] J. Curtius. Nucleation of atmospheric aerosol particles. *Comptes Rendus Physique*, 7(9-10):1027–1045, NOV-DEC 2006.

[47] M. Noppel, H. Vehkamaki, and M. Kulmala. An improved model for hydrate formation in sulfuric acid-water nucleation. *J. Chem. Phys.*, 116(1):218–228, JAN 1 2002.

[48] H. Vehkamaki, M. Kulmala, I. Napari, K.E.J. Lehtinen, C. Timmreck, M. Noppel, and A. Laaksonen. An improved parameterization for sulfuric acid-water nucleation rates for tropospheric and stratospheric conditions. *Journal Of Geophysical Research-Atmospheres*, 107(D22), NOV 2002.

[49] F.Q. Yu. Binary H2SO4-H2O homogeneous nucleation based on kinetic quasi-unary nucleation model: Look-up tables. *Journal Of Geophysical Research-Atmospheres*, 111(D4), FEB 17 2006.

[50] D. R. Hanson and E. R. Lovejoy. Measurement of the thermodynamics of the hydrated dimer and trimer of sulfuric acid. *J. Phys. Chem. A*, 110(31):9525–9528, AUG 10 2006.

[51] D.J. Coffman and D.A. Hegg. A preliminary-study of the effect of ammonia on particle nucleation in the marine boundary-layer. *Journal Of Geophysical Research-Atmospheres*, 100(D4):7147–7160, APR 20 1995.

[52] R.J. Weber, J.J. Marti, P.H. McMurry, F.L. Eisele, D.J. Tanner, and A. Jefferson. Measured atmospheric new particle formation rates: Implications for nucleation mechanisms. *Chemical Engineering Communications*, 151:53–64, 1996.

[53] R.Y. Zhang, I. Suh, J. Zhao, D. Zhang, E.C. Fortner, X.X. Tie, L.T. Molina, and M.J. Molina. Atmospheric new particle formation enhanced by organic acids. *SCIENCE*, 304(5676):1487–1490, JUN 4 2004.

[54] C. Hoose and O. Möhler. Heterogeneous ice nucleation on atmospheric aerosols: a review of results from laboratory experiments. *Atmos. Chem. Phys.*, 12(20):9817–9854, 2012.

[55] H. R. Pruppacher, J. D. Klett, and P. K. Wang. Microphysics of clouds and precipitation. 1998.

[56] B. J. Mason. The physics of clouds, clarendon, 1971.

[57] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *J. Chem. Phys.*, 21(6):1087–1092, 1953.

[58] S. Dietrich and M. Schick. Critical Wetting of Surface in Systems with Long-Range Forces. *Phys. Rev. B.*, 31(7):4718–4720, 1985.

[59] K. Kaneda and Y. Okabe. Finite-size scaling for the Ising model on the Möbius strip and the Klein bottle. *Phys. Rev. Lett.*, 86(10):2134, 2001.

[60] Pixar Animation Studios. The RenderMan Interface Specification. v3.0.

[61] J. A. Anderson, C. D. Lorenz, and A. Travesset. General purpose molecular dynamics simulations fully implemented on graphics processing units. *J. Comput. Phys.*, 227:5342–5359, 2008.

[62] M. S. Friedrichs, P. Eastman, V. Vaidyanathan, M. Houston, S. Legrand, A. L. Beberg, D. L. Ensign, C. M. Bruns, and V. S. Pande. Accelerating molecular dynamic simulation on graphics processing units. *J. Comput. Chem.*, 30:864–872, 2009.

[63] D. Reith, A. Milchev, P. Virnau, and K. Binder. Anomalous structure and scaling of ring polymer brushes. *Eur. Phys. Let.*, 95, 2011.

[64] D. Reith, L. Mirny, and P. Virnau. GPU based molecular dynamics simulations of polymer rings in concentrated solution: Structure and scaling. *Prog. Theor. Phys. Supp.*, pages 135–145, 2011.

[65] van Meel J. A., Arnold A., and Frenkel D. Harvesting graphics power for md simulations. *Mol. Simulat.*, 34:259–266, 2008.

[66] W. Xian and A. Takayuki. Multi-GPU performance of incompressible flow computation by lattice Boltzmann method on GPU cluster. *Parallel Comput.*, 37(9, SI):521–535, SEP 2011.

[67] K. R. Tubbs and F. T. C. Tsai. GPU accelerated lattice Boltzmann model for shallow water flow and mass transport. *Int. J. Numer. Meth. Eng.*, 86(3):316–334, APR 22 2011.

[68] T. Preis, P. Virnau, W. Paul, and J. J. Schneider. GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *J. Comput. Phys.*, 228:4468–4477, 2009.

[69] B. Huang, J. Mielikainen, H. Oh, and H.-L. A. Huang. Development of a GPU-based high-performance radiative transfer model for the Infrared Atmospheric Sounding Interferometer (IASI). *J. Comput. Phys.*, 230(6):2207–2221, MAR 20 2011.

[70] J. Schmidt, C. Piret, N. Zhang, B. J. Kadlec, D. A. Yuen, Y. Liu, G. B. Wright, and Erik O. D. Sevre. Modeling of tsunami waves and atmospheric swirling flows with graphics processing unit (GPU) and radial basis functions (RBF). *Concurr. Comp.-Pract. E*, 22(12, SI):1813–1835, AUG 25 2010.

[71] K.S. Oh and K. Jung. GPU implementation of neural networks. *Pattern Recogn.*, 37(6):1311–1314, JUN 2004.

[72] P. D. Vouzis and N. V. Sahinidis. GPU-BLAST: using graphics processors to accelerate protein sequence alignment. *Bioinformatics*, 27(2):182–188, JAN 15 2011.

[73] B. J. Block. Platform independent, efficient implementation of the Ising model on parallel acceleration devices. *Eur. Phys. J. Spec. Top*, 2012.

[74] B. J. Block and T. Preis. Computer simulations of the Ising model on graphics processing units. *Eur. Phys. J. Spec. Top*, 2012.

[75] B. J. Block. Diploma thesis. 2011.

[76] B. Block, P. Virnau, and T. Preis. Multi-GPU accelerated multi-spin Monte Carlo simulations of the 2D Ising model. *Comput. Phys. Commun.*, 181(9):1549–1556, September 2010.

[77] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krueger, A. E. Lefohn, and T. J. Purcell. A survey of general-purpose computation on graphics hardware. *Comput. Graph. Forum*, 26(1):80–113, 2007.

[78] S. Tomov, M. McGuigan, R. Bennett, G. Smith, and J. Spiletic. Benchmarking and implementation of probability-based simulations on programmable graphics cards. *Computers & Graphics*, 29:71–80, 2005.

[79] J. Nickolls and W. J. Dally. The GPU Computing Era. *IEEE Micro*, 30(2):56–69, MAR-APR 2010.

[80] P. Messmer, P.J. Mullowney, and B.E. Granger. GPULIB: GPU computing in high-level languages. *Comput. Sci. Eng.*, pages 70–73, 2008.

[81] J. Hoberock and N. Bell. Thrust: A parallel template library, 2010. Version 1.3.0.

[82] T. B. Jablin, P. Prabhu, J. A. Jablin, N. P. Johnson, S. R. Beard, and D. I. August. Automatic CPU-GPU Communication Management and Optimization. *ACM SIGPLAN Notices*, 46(6):142–151, JUN 2011.

[83] C.-T. Hong, D.-H. Chen, Y.-B. Chen, W.-G. Chen, W.-M. Zheng, and H.-B. Lin. Providing Source Code Level Portability Between CPU and GPU with MapCG. *J. Comput. Sci. Technol.*, 27(1):42–56, JAN 2012.

[84] D. Chen, W. Chen, and W. Zheng. CUDA-Zero: a framework for porting shared memory GPU applications to multi-GPUs. *Sci. China - Inf. Sci.*, 55(3):663–676, MAR 2012.

[85] J. Fang, A. L. Varbanescu, and H. Sips. A comprehensive performance comparison of cuda and opencl. In *The 40-th International Conference on Parallel Processing (ICPP'11), Taipei, Taiwan*, September 2011.

[86] G. Diamos, A. Kerr, and M. Kesavan. Translating GPU binaries to tiered simd architectures with ocelot. Technical Report GIT-CERCS-09-01, Georgia Institute of Technology, January 2009.

[87] NVIDIA Corporation. *NVIDIA CUDA C Programming Guide*, 2013. Version 5.5.

[88] M. Weigel. Performance potential for simulating spin models on GPU. *J. Comp. Phys.*, 231(8):3064–3082, APR 20 2012.

[89] M. Weigel. Simulating spin models on GPU. *Comp. Phys. Comm.*, 182(9, SI):1833–1836, SEP 2011.

[90] M. Bernaschi, M. Fatica, G. Parisi, and L. Parisi. Multi-GPU codes for spin systems simulations. *Comp. Phys. Comm.*, 183:1416–1421, 2012.

[91] F. Lu, J. Song, F. Yin, and X. Zhu. Performance evaluation of hybrid programming patterns for large CPU/GPU heterogeneous clusters. *Comput. Phys. Commun.*, 183:1172–1181, 2012.

[92] Y. Komura and Y. Okabe. GPU-based single-cluster algorithm for the simulation of the Ising model. *J. Comput. Phys*, 231(4):1209–1215, February 2012.

[93] Y. Komura and Y. Okabe. GPU-based Swendsen-Wang multi-cluster algorithm for the simulation of two-dimensional classical spin systems. *Comp. Phys. Comm.*, 183(6):1155–1161, JUN 2012.

[94] N. Ito and Y. Kanada. An Effective Algorithm for the Monte-Carlo Simulation of the Ising-Model on a Vector Processor. *Supercomputer*, 5:31, 1988.

[95] U. Wolff. Collective Monte-Carlo Updating For Spin Systems. *Phys. Rev. Let.*, 62(4):361–364, JAN 23 1989.

[96] R.H. Swendsen and J.S. Wang. Nonuniversal Critical-Dynamics In Monte-Carlo Simulations. *Phys. Rev. Let.*, 58(2):86–88, JAN 12 1987.

[97] J. Kaupuzs, J. Rimsans, and R. V. N. Melnik. Parallelization of the Wolff single-cluster algorithm. *Phys. Rev. E*, 81(2, Part 2), FEB 2010.

[98] J. Hoshen and R. Kopelman. percolation And Cluster Distribution .1. Cluster Multiple Labeling Technique And Critical Concentration Algorithm. *Phys. Rev. B*, 14(8):3438–3445, 1976.

[99] M. Matsumoto, M. Saito, H. Haramoto, and T. Nishimura. Pseudorandom number generation: Impossibility and compromise. *J. Univ. Comp. Sci.*, 12(6):672–690, 2006.

[100] M. Manssen, M. Weigel, and A. K. Hartmann. Random number generators for massively parallel simulations on GPU. *European Physical Journal-Special Topics*, 210(1):53–71, AUG 2012.

[101] J. J. Schneider and S. Kirkpatrick. *Stochastic Optimization*. Springer, Berlin, 2006.

[102] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 2007.

[103] H. Nguyen. *GPU Gems 3*. Addison-Wesley Professional, August 2007.

[104] R.C. Tausworthe. Random numbers generated by linear recurrence modulo two. *Math. Comput.*, pages 201–209, 1965.

[105] S. Wansleben, J. G. Zabolitzky, and C. Kalle. Monte Carlo Simulation of Ising Models by Multispin Coding on a Vector Computer. *J. Stat. Phys.*, 37:271–282, 1984.

[106] R. Zorn, H. J. Herrmann, and C. Rebbi. Parallelization of the Ising simulation. *Comput. Phys. Commun.*, 23:337–342, 1981.

[107] N. Ito and Y. Kanada. Monte Carlo Simulation of the Ising Model and Random Number Generation on the Vector Processor. *Supercomputing 90*, 1990.

[108] A.T. Ogielski. Dynamics of fluctuations in the ordered phase of kinetic Ising-models. *Physical Review B*, 36(13):7315–7318, NOV 1 1987.

[109] D. Stauffer. Relaxation of Ising models near and away from criticality. *Physica A*, 244(1-4):344–357, OCT 1 1997.

[110] B. J. Block, M. Lukáčová-Medvid'ová, P. Virnau, and L. Yelash. Accelerated GPU simulation of compressible flow by the discontinuous evolution Galerkin method. *European Physical Journal-Special Topics*, 210(1):119–132, AUG 2012.

[111] B. J. Block, S. K. Das, M. Oettel, P. Virnau, and K. Binder. Curvature dependence of surface free energy of liquid drops and bubbles: A simulation study. *J. Chem. Phys.*, 133(15), OCT 21 2010.

[112] R. A. Fisher and F. et. al. Yates. Statistical tables for biological, agricultural and medical research. *Statistical tables for biological, agricultural and medical research.*, (Ed. 3.), 1949.

[113] E. Bittner, A. Nussbaumer, and W. Janke. Anisotropy of the interface tension of the three-dimensional Ising model. *Nucl.Phys.*, B820:694–706, 2009.

[114] M. Hasenbusch and K. Pinn. Comparison of Monte Carlo results for the 3-d Ising interface tension and interface energy with (extrapolated) series expansions. *Physica*, A203:189–213, 1994.

[115] P.E. Wolf, F. Gallet, S. Balibar, E. Rolley, and P. Nozieres. Crystal-Growth and Crystal Curvature near Roughening Transitions in HCP HE-4. *Journal de Physique*, 46(11):1987–2007, 1985.

[116] A. Hundertmark-Zauskova, M. Lukáčová-Medvid'ová, and F. Prill. Large Time Step Finite Volume Evolution Galerkin Methods. *J. Sc. Comp.*, 48(1-3):227–240, JUL 2011.

[117] M. Lukáčová-Medvid'ová, G. Warnecke, and Y. Zahaykah. On the boundary conditions for EG methods applied to the two-dimensional wave equation system. *Zeitschrift für angewandte Mathematik und Mechanik*, 84(4):237–251, 2004.

[118] M. Lukáčová-Medvid'ová, S. Noelle, and M. Kraft. Well-balanced finite volume evolution Galerkin methods for the shallow water equations. *J. Comp. Phys.*, 221(1):122–147, JAN 20 2007.

[119] L. Yelash, A. Müller, M. Lukáčová-Medvid'ová, F.X. Giraldo, and V. Wirth. *J. Comp. Phys.*, 268:106–133, 2014.

[120] F. X. Giraldo, M. Restelli, and M. Laeuter. Semi-implicit formulations of the Navier-Stokes equations: Application to nonhydrostatic atmospheric modeling. *Siam. Journal on Scientific Computing*, 32(6):3394–3425, 2010.

[121] A. Müller, J. Behrens, F. X. Giraldo, and V. Wirth. Testing refinement criteria in adaptive discontinuous Galerkin simulations of dry atmospheric convection. Technical report, DTIC Document, 2011.

[122] J. M. Straka, R.B. Wilhelmson, L.J. Wicker, J.R. Anderson, and K.K. Droegemeier. Numerical-Solutions of a nonlinear density-current - A Benchmark solution and comparisons. *International Journal For Numerical Methods In Fluids*, 17(1):1–22, JUL 15 1993.

[123] R. J. LeVeque. *Finite volume methods for hyperbolic problems*, volume 31. Cambridge university press, 2002.

[124] E. F. Toro. *Riemann solvers and numerical methods for fluid dynamics: a practical introduction*. Springer, 2009.

[125] L. C. Wilcox, G. Stadler, C. Burstedde, and O. Ghattas. A high-order discontinuous Galerkin method for wave propagation through coupled elastic–acoustic media. *J. Comput. Phys.*, 229(24):9373–9396, 2010.

[126] J. F. Kelly and F. X. Giraldo. Continuous and discontinuous Galerkin methods for a scalable three-dimensional nonhydrostatic atmospheric model: Limited-area mode. *J. Comput. Phys.*, 231(24):7988–8008, 2012.

[127] M. Restelli and F. X. Giraldo. A Conservative Discontinuous Galerkin Semi-implicit Formulation For The Navier-stokes Equations In Nonhydrostatic Mesoscale Modeling. *Siam Journal On Scientific Computing*, 31(3):2231–2257, 2009.

[128] A. Müller, J. Behrens, F. X. Giraldo, and V. Wirth. An adaptive discontinuous Galerkin model for modeling cumulus clouds. *Proceedings of the V European Conference on Computational Fluid Dynamics ECCOMAS CFD*, 2010.

[129] F.X. Giraldo, J.S. Hesthaven, and T. Warburton. Nodal high-order discontinuous Galerkin methods for the spherical shallow water equations. *J. Comput. Phys.*, 181(2):499–525, 2002.

[130] F.X. Giraldo and T. Warburton. A high-order triangular discontinuous galerkin oceanic shallow water model. *International journal for numerical methods in fluids*, 56(7):899–925, 2008.

[131] J. Qiu, B. C. Khoo, and C.-W. Shu. A numerical study for the performance of the Runge–Kutta discontinuous Galerkin method based on different numerical fluxes. *J. Comput. Phys.*, 212(2):540–565, 2006.

[132] M. Lukáčová-Medvid'ová, K.W. Morton, and G. Warnecke. Evolution Galerkin methods for hyperbolic systems in two space dimensions. *Mathematics of Computation*, 69(232):1355–1384, OCT 2000.

[133] J. Behrens, N. Rakowsky, W. Hiller, D. Handorf, M. Lauter, J. Papke, and K. Dethloff. amatos: Parallel adaptive mesh generator for atmospheric and oceanic simulation. *Ocean Modelling*, 10(1-2):171–183, 2005.

[134] F. X. Giraldo and M. Restelli. A study of spectral element and discontinuous Galerkin methods for the Navier-Stokes equations in nonhydrostatic mesoscale atmospheric modeling: Equation sets and test cases. *J. Comp. Phys.*, 227(8):3849–3877, APR 1 2008.

[135] A. Robert. Bubble convection experiments with a semi-implicit formulation of the Euler equations. *J. Atmos. Sci.*, 50(13):1865–1873, 1993.

Declaration

I hereby declare that I wrote the dissertation submitted without any unauthorized external assistance and used only sources acknowledged in the work.

All textual passages which are appropriated verbatim or paraphrased from published and unpublished texts as well as all information obtained from oral sources are duly indicated and listed in accordance with bibliographical rules.

In carrying out this research, I complied with the rules of standard scientific practice as formulated in the statutes of Johannes Gutenberg-University Mainz to insure standard scientific practice.

Benjamin J. Block