HZDR

HELMHOLTZ
ZENTRUM DRESDEN
ROSSENDORF

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Efficient Parallel Monte-Carlo Simulations for Large-Scale Studies of Surface Growth Processes

Von der Fakultät für Naturwissenschaften der
Technischen Universität Chemnitz

genehmigte

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium
(Dr. rer. nat.)

vorgelegt

von

Dipl. Phys. Jeffrey Kelling

geboren am 7. Februar 1987 in Rüdersdorf

Tag der Einreichung 14.06.2017

Gutachter:
Prof. Dr. Sibylle Gemming
Dr. Martin Weigel

Tag der Verteidigung 13.04.2018

Dissertation
Jeffrey Kelling,
Efficient Parallel Monte-Carlo Simulations for Large-Scale Studies of Surface Growth Processes,
Technische Universität Chemnitz, Faculty of Natural Sciences (2017),
142 pages, 34 figures, 10 tables
Typeset on 13th August 2018

Keywords: lattice Monte Carlo, surface growth, ballistic deposition, physical aging, Kardar–Parisi–Zhang-equation, lattice gas, parallel processing, domain decomposition, GPU

Abstract

Lattice Monte Carlo methods are used to investigate far from and out-of-equilibrium systems, including surface growth, spin systems and solid mixtures. Applications range from the determination of universal growth or aging behaviors to palpable systems, where coarsening of nanocomposites or self-organization of functional nanostructures are of interest. Such studies require observations of large systems over long times scales, to allow structures to grow over orders of magnitude, which necessitates massively parallel simulations.

This work addresses the problem of parallel processing introducing correlations in Monte Carlo updates and proposes a virtually correlation-free domain decomposition scheme to solve it. The effect of correlations on scaling and dynamical properties of surface growth systems and related lattice gases is investigated further by comparing results obtained by correlation-free and intrinsically correlated but highly efficient simulations using a stochastic cellular automaton (SCA). Efficient massively parallel implementations on graphics processing units (GPUs) were developed, which enable large-scale simulations leading to unprecedented precision in the final results.

The primary subject of study is the Kardar–Parisi–Zhang (KPZ) surface growth in $(2+1)$ dimensions, which is simulated using a dimer lattice gas and the restricted solid-on-solid model (RSOS) model. Using extensive simulations, conjectures regarding growth, autocorrelation and autoresponse properties are tested and new precise numerical predictions for several universal parameters are made.

Dissertation
Jeffrey Kelling,
Efficient Parallel Monte-Carlo Simulations for Large-Scale Studies of Surface Growth Processes,
Technische Universität Chemnitz, Fakultät für Naturwissenschaften (2017),
142 Seiten, 34 Abbildungen, 10 Tabellen
Gesetzt am 13. August 2018

Stichworte: Gitter-Monte-Carlo-Simulation, Oberfächenwachstum, Teilchenabscheidung, Alterung, Kardar–Parisi–Zhang-Gleichung, Gittergas, Paralleles Programm, Domänenzerlegung, GPU

Zusammenfassung

Gitter-Monte-Carlo-Methoden werden zur Untersuchung von Systemen wie Oberflächenwachstum, Spinsystemen oder gemischten Feststoffen verwendet, welche fern eines Gleichgewichtes bleiben oder zu einem streben. Die Anwendungen reichen von der Bestimmung universellen Wachstums- und Alterungsverhaltens hin zu konkreten Systemen, in denen die Reifung von Nanokompositmaterialien oder die Selbstorganisation von funktionalen Nanostrukturen von Interesse sind. In solchen Studien müssen große Systemen über lange Zeiträume betrachtet werden, um Strukturwachstum über mehrere Größenordnungen zu erlauben. Dies erfordert massivparallele Simulationen.

Diese Arbeit adressiert das Problem, dass parallele Verarbeitung Korrelationen in Monte-Carlo-Updates verursachen und entwickelt eine praktisch korrelationsfreie Domänenzerlegungsmethode, um es zu lösen. Der Einfluss von Korrelationen auf Skalierungs- und dynamische Eigenschaften von Oberflächenwachtums- sowie verwandten Gittergassystemen wird weitergehend durch den Vergleich von Ergebnissen aus korrelationsfreien und intrinsisch korrelierten Simulationen mit einem stochastischen zellulären Automaten untersucht. Effiziente massiv parallele Implementationen auf Grafikkarten wurden entwickelt, welche großskalige Simulationen und damit präzedenzlos genaue Ergebnisse ermöglichen.

Das primäre Studienobjekt ist das $(2+1)$-dimensionale Kardar–Parisi–Zhang-Oberflächenwachstum, welches durch ein Dimer-Gittergas und das Kim-Kosterlitz-Modell simuliert wird. Durch massive Simulationen werden Thesen über Wachstums-, Autokorrelations- und Antworteigenschaften getestet und neue, präzise numerische Vorhersagen zu einigen universellen Parametern getroffen.

Contents

1. Introduction **9**
- 1.1. Motivations and Goals 10
- 1.2. Overview 11

2. Methods and Models **13**
- 2.1. Estimation of Scaling Exponents and Error Margins 13
- 2.2. From Continuum- to Atomistic Models 16
- 2.3. Models for Phase Ordering and Nanostructure Evolution 18
  - 2.3.1. The Kinetic Metropolis Lattice Monte-Carlo Method 19
  - 2.3.2. The Potts Model 19
- 2.4. The Kardar–Parisi–Zhang and Edwards–Wilkinson Universality Classes 21
    - 2.4.0.1. Physical Aging 23
  - 2.4.1. The Octahedron Model 23
  - 2.4.2. The Restricted Solid on Solid Model 25

3. Parallel Implementation: Towards Large-Scale Simulations **27**
- 3.1. Parallel Architectures and Programming Models 29
  - 3.1.1. CPU 30
  - 3.1.2. GPU 31
  - 3.1.3. Heterogeneous Parallelism and MPI 33
  - 3.1.4. Bit-Coding of Lattice Sites 34
- 3.2. Domain Decomposition for Stochastic Lattice Models 34
  - 3.2.1. DD for Asynchronous Updates 35
    - 3.2.1.1. Dead border (DB) 35
    - 3.2.1.2. Double tiling (DT) 37
    - 3.2.1.3. DT DD with random origin (DTr) 37
    - 3.2.1.4. Implementation 37
  - 3.2.2. Second DD Layer on GPUs 38
    - 3.2.2.1. Single-Hit DT 39
    - 3.2.2.2. Single-Hit dead border (DB) 39
    - 3.2.2.3. DD Parameters for the Octahedron Model 39
  - 3.2.3. Performance 40
- 3.3. Lattice Level DD: Stochastic Cellular Automaton 40
  - 3.3.1. Local Approach for the Octahedron Model 41
  - 3.3.2. Non-Local Approach for the Octahedron Model 41
    - 3.3.2.1. Bit-Vectorized GPU Implementation 41
  - 3.3.3. Performance of SCA Implementations 44

3.4. The Multi-Surface Coding Approach 46
    3.4.0.1. Vectorization 47
    3.4.0.2. Scalar Updates 48
    3.4.0.3. Domain Decomposition 49
  3.4.1. Implementation: SkyMC 50
    3.4.1.1. 2d Restricted Solid on Solid Model 50
    3.4.1.2. 2d and 3d Potts Model 51
    3.4.1.3. Sequential CPU Reference 52
  3.4.2. SkyMC Benchmarks 52
3.5. Measurements 55
    3.5.0.1. Measurement Intervals 55
    3.5.0.2. Measuring using Heterogeneous Resources 56

4. Monte-Carlo Investigation of the Kardar–Parisi–Zhang Universality Class 57
4.1. Evolution of Surface Roughness 58
  4.1.1. Comparison of Parallel Implementations of the Octahedron Model 59
    4.1.1.1. The Growth Regime 60
    4.1.1.2. Distribution of Interface Heights in the Growth Regime 64
    4.1.1.3. KPZ Ansatz for the Growth Regime 65
    4.1.1.4. The Steady State 68
  4.1.2. Investigations using RSOS 69
    4.1.2.1. The Growth Regime 69
    4.1.2.2. The Steady State 72
    4.1.2.3. Consistency of Fine-Size Scaling with Respect to DD 75
  4.1.3. Results for Growth Phase and Steady State 76
4.2. Autocorrelation Functions 78
  4.2.1. Comparison of DD Methods for RS Dynamics 78
    4.2.1.1. Device-Layer DD 78
    4.2.1.2. Block-Layer DD 80
  4.2.2. Autocorrelation Properties under RS Dynamics 83
  4.2.3. Autocorrelation Properties under SCA Dynamics 84
    4.2.3.1. Autocorrelation of Heights 87
    4.2.3.2. Autocorrelation of Slopes 89
  4.2.4. Autocorrelation in the SCA Steady State 89
  4.2.5. Autocorrelation in the EW Case under SCA 91
    4.2.5.1. Autocorrelation of Heights 91
    4.2.5.2. Autocorrelations of Slopes 92
4.3. Autoresponse Functions 94
  4.3.1. Autoresponse Properties 95
    4.3.1.1. Autoresponse of Heights 95
    4.3.1.2. Autoresponse of Slopes 97
    4.3.1.3. Self-Averaging 97
4.4. Summary 98

5. Further Topics **101**
- 5.1. Investigations of the Potts Model 101
  - 5.1.1. Testing Results from the Parallel Implementations 101
  - 5.1.2. Domain Growth in Disordered Potts Models 102
- 5.2. Local Scale Invariance in KPZ Surface Growth 105

6. Conclusions and Outlook **109**

Acknowledgements **114**

A. Coding Details **117**
- A.1. Bit-Coding 117
- A.2. Packing and Unpacking Signed Integers 117
- A.3. Random Number Generation 118

Bibliography **120**

List of Figures **133**

List of Tables **134**

Glossary **135**

Acronyms **136**

Selbstständigkeitserklärung **138**

Curriculum Vitæ **139**

List of Publications (Kelling, J.) **140**

1. Introduction

Monte Carlo (MC) simulations have first been developed in the Manhattan project to calculate the statistics of neutron flux [1–3] and were later adapted for many problems in statistical physics [4] and beyond. Kinetic MC simulations are especially useful for studying the evolution of non-equilibrium systems including nano-structures [5,6] and surfaces [7]. The transferability of the employed stochastic models' properties to various physical systems is provided by scale-invariance and *universality* of the defining stochastic processes.

The stochastic microscopic dynamics of particles can give rise to a macroscopic evolution of system properties and structures when a system is out of equilibrium. When the resulting kinetics does not depend on the dynamics found in a specific system, which includes scale-invariance, the process is considered universal. The notion of universality was first introduced by the discovery, that critical exponents in quite different systems are equal near second order phase transitions and are less sensitive to the details of the system the closer they are to the critical point [8]. It was also found to apply in non-equilibrium systems to phase-ordering [9] and relaxation [10] phenomena as well as interface growth [11]. Such processes unfold at criticality and exhibit universal dynamical and growth exponents.

Many stochastic processes lead to the formation of regular structures, for example, processes of relaxation such as coarsening after phase separation, including Ostwald ripening [12] and coarsening after spinodal decomposition [13], or the Plateau–Rayleigh instability [14, 15], which are based on diffusive dynamics and are driven by surface tension. These examples can be realized in many different systems, like liquid or solid mixtures, even the universe as a whole [16], thereby providing graphic examples of scale-invariant universal processes. More effects can be observed in the presence of competing driving forces, for instance inverse Ostwald ripening [17, 18] or the formation of ripples on a surface [19]. Comparable surface patterns can form at any phase boundary, including solid–solid and vapor–gas or surfaces of solids.

Such far from or out-of equilibrium processes can be studied best in large-scale atomistic simulations, which require large amounts of computing power. In the past, the sequential performance of compute units was steadily increasing. This was mainly driven by the reduction of the size of circuit components in modern integrated electronics. Smaller components, such as narrower channels in transistors, allow larger circuits to fit onto small chips, which enables higher integration of electronic devices and subsequently lowers the production cost of more complex devices. Integrating more units on a chip, for instance more cores or larger caches for compute devices, increases performance, not least by shortening signal paths between these resources. The main advantage of small circuit components is their reduced energy consumption, which is the main limiting factor of the achievable clock frequency and thus of the performance of individual components.

The famous "Moore's law" [20], formulated in 1965, states that the number of transistors on a single chip doubles every 18 months as a result of progress in fabrication technologies. With latest processors being produced using 14 nm fabrication processes, lithography techniques have already gone far beyond the limits set by the wavelength of visible light which is achieved by etching structures to smaller sizes. Producing nanostructures *top-down* becomes more demanding the smaller the desired structures are. Current developments in lithography techniques involve UV-light or electron beams. Furthermore, at the achieved structure sizes, there is little margin left for further downscaling, since bulk materials become discrete not much further down (the lattice constant of silicon is about 0.5 nm). New approaches involve stacking components into the third dimension, making production processes more complex, and thus even more demanding. Processes for the deposition of material (i. e. particles) or removal of the same (etching) are stochastic and the smaller the relevant scales are, the more significant random fluctuations become, which then need to be controlled.

With the growth of *sequential* compute power stagnating, *parallel* compute resources provide the only path forward for simulations. In the case of MC this paradigm can be difficult to implement, because parallel execution of an algorithm changes the dynamics of the simulated stochastic process and this artificial dynamics may lead to artifacts in the result. This is especially troublesome, since simulating effects of self-organisation is helpful for the development of *bottom-up* fabrication techniques for nano- or microstructures used in integrated circuits [21, 22]. At the same time, such simulations are also used to study the aging of materials, which is governed by similar processes, for example in porous catalysts [23]. All of these results could become biased when parallel simulations are used.

1.1. Motivations and Goals

This work is centered around computer experiments based on lattice MC simulations. One focus lies on the technique itself, with the goal of enabling accurate simulations which are scalable through parallelization. Molecular dynamics (MD) and methods relying on the numerical integration of differential equations can be parallelized exactly, because they intrinsically work on vectors of particles or elements, respectively. For MC algorithms, parallel implementations can only be approximations to their actually sequential definitions. Thus, the main questions are if and how the artificial dynamics in parallel simulations affects the kinetics of evolution in stochastic lattice models. To this end different approaches are compared and a parallel implementation is presented which can serve as a virtually identical replacement for sequential simulations. The parallel implementations presented in this work are primarily designed to run on single graphics processing units (GPUs), which provide a better ratio of compute performance over power consumption than central processing units

(CPUs). Thereby, using GPUs enables more extensive simulations consuming less energy.

The other main focus of this work lies on surface growth by ballistic deposition of particles in the Kardar–Parisi–Zhang (KPZ) universality class. The underlying stochastic process does not include explicit thermally activated dynamics, which makes it very sensitive to potentially disturbed dynamics introduced by a parallel implementation. Thus, while this system is not one that exhibits self-organization, it provides a better benchmark for parallelization schemes. Furthermore, the model employed in the presented surface growth simulations is based on an underlying lattice gas of two particle species, the kinetics of which provide yet another test case. Results of extensive large-scale simulations, employing the methods developed here, provide predictions of unprecedented accuracy for scaling and aging properties of systems in the KPZ and Edwards–Wilkinson (EW) universality classes.

All of the above is also meant to enable MC simulations of atomistic models at spatio-temporal scales accessed in experiments in solid-state physics, such as the initially mentioned bottom-up processes for device-fabrication or aging nano- or microstructures. Here too, trade-offs can be made between computational efficiency and accuracy of statistics by choosing between the different types of parallel MC simulations presented here.

1.2. Overview

This work is structured as follows:

In chapter 2 the stochastic models and numerical methods employed in this work are introduced. Section 2.1 describes numerical methods to deal with simulation results, which shall serve to illustrate the necessity of the parallel large-scale simulations for the sake of reliable numerical estimates, rather than just to enable experimental-scale simulation. The basic ideas of the MC method and the Metropolis algorithm are introduced in section 2.2. Section 2.3 describes models for phase ordering and domain growth, focussing on the Potts model. This mainly serves to embed the work in the larger scope of simulating solid phases, or solid solution, using MC simulations. Potts model simulations are the subject of section 5.1.

Chapter 4 is dedicated to extensive Monte Carlo studies of the $2+1$–dimensional KPZ universality class, with a short excursion into the EW class in section 4.2.5. The evolution of the surface roughness and steady state properties is investigated using both the octahedron model and the restricted solid-on-solid model (RSOS) (section 4.1). The most extensive simulations presented are to investigate physical aging in the octahedron model, by precisely determining the autocorrelation and autoresponse functions (sections 4.2 and 4.3).

In both areas, a significant part of the study is dedicated to the analysis of the effects of parallel implementations on the results obtained. In this context a parallelization method for random-sequential (RS) Monte Carlo updates without measurable correlation is developed (sections 3.2 and 4.2.1). Some cases have been identified where correlated updates using a stochastic cellular automaton (SCA) scheme (section 3.3) give correct results and thus enable vastly more efficient simulations. The developed parallel implementations are described in chapter 3.

With the development and evaluation of parallel implementations for lattice MC being one focus area of this work, the applicability of the developed methods to a wide range of problems is important. With this in mind, aside from the work regarding the KPZ universality class, presented in chapter 4, exploratory studies of a few further topics have been performed. Chapter 5 presents preliminary results for two such studies, predominantly with the purpose of demonstrating the potential for wider applications in the methods described in chapter 3.

Finally, a summary and a list of the main theses of the present work are provided in chapter 6.

2. Methods and Models

This chapter introduces the models considered in this work, the methods used to obtain numerical results as well as connected challenges. Section 2.1 starts by describing, in a general manner, the main methods used in this work to derive estimates of universal exponents in the investigated systems. It illustrates some challenges towards obtaining reliable numerical estimates and thereby motivates the necessity of highly precise computer-experimental results.

Sections 2.2 and 2.3 remind the reader of the most important ideas behind Monte Carlo (MC) methods and introduce models for the evolution of nanostructures alongside the Metropolis algorithm. Finally, section 2.4 introduces the surface growth systems and models which are investigated in chapter 4, making up the main part of the present work.

2.1. Estimation of Scaling Exponents and Error Margins

All models investigated in this work are stochastic, thus there is always intrinsic noise in the simulation. Even directly integrating a stochastic differential equation numerically would not immediately yield the expectation value for any observable. The result of a single simulation only represents a single random sample, thus many independent runs need to be averaged to yield a reliable estimate for the expectation value of any observable f:

$$\langle f \rangle = \sum_i^N f_i / N \quad , \tag{2.1}$$

where N is the number of samples averaged over. The standard error estimator is given as:

$$\Delta f = \sqrt{\frac{1}{N(N-1)} \cdot \sum_i^N \left(f_i - \langle f \rangle\right)^2} \quad . \tag{2.2}$$

If the f_i are normally distributed, one can derive the usual confidence intervals, but not all random variables encountered in this work follow a pure normal distribution.

If the expectation value of the observable was the desired result, the estimation of error margins would not require any further discussion, but none of the final results to be obtained in this work are simply the expectation values of observables. Most of the quantities of interest are dynamical properties of an observable which is a function of time $f(t)$ and, in critical systems, follows a power law (PL) [11,24,25]:

$$f(t) \sim t^{\gamma} \quad . \tag{2.3}$$

The exponent γ is universal for the investigated class of systems and shall be the quantity of interest here. This form could be fitted to the data obtained from simulations: $\langle f(t)\rangle \pm \Delta f(t)$, where a statistical error $\Delta\gamma$ can be computed. Note that, equation 2.3 gives only the asymptotic form of $f(t)$ for $t \to \infty$. The form of $f(t)$ at finite times may have other contributions. These include non-universal corrections which are properties of the specific system under investigation rather than universal ones. In some universality classes the scaling functions are known to take forms like $f(t) \sim g(t)^{\gamma}$, but the exact form of $g(t)$ is only known for some analytically solved models.

Corrections to the scaling function can be caused by fixed points of other universality classes, which are near-by in the sense of renormalization group theory [26]. In many cases, these are known to introduce PL or logarithmic corrections. Thus to a first order it is a good approach to assume PL corrections. It is often necessary to determine these corrections empirically, when their origin is not known.

One type of corrections, on the other hand, usually has a quite intuitive origin: Additive constant corrections, which can be subtracted from the data before further analysis:

$$\langle f(t)\rangle = \langle f'(t)\rangle - o \quad .$$

Possible sources of which include discretization or correlations in the model used. The simplest example is the intrinsic width of the surface in the octahedron model expressed in the roughness of a flat surface, giving $o = W_{\text{flat}} = 0.25$ [27].

For a stochastic model the form of $f(t)$ may well be very sensitive to the specific type of dynamics employed in the simulation. In a double logarithmic plot the data $\langle f(t)\rangle$ may even appear to resemble a perfectly straight line, but when fitting (2.3) to a different interval of the data, the resulting exponent γ will change. This can be quantified by computing the effective exponents at different times t:

$$\gamma_{\text{eff}}\left(t = (t_n - t_m)/2\right) = \frac{\ln\left(\langle f(t_n)\rangle\right) - \ln\left(\langle f(t_m)\rangle\right)}{\ln t_n - \ln t_m} \quad , \tag{2.4}$$

where $t_m < t_n$. If $f(t)$ followed a pure power-law, $\gamma_{\text{eff}}(t)$ would be constant. Here, the choice of t_n and t_m will ideally not change the result, though in case $\langle f(t)\rangle$ is very noisy, a large $t_n - t_m$ may be more likely to give a useful curve of effective exponents. A point-wise error $\Delta\gamma_{\text{eff}}(t)$ can be computed by Gaussian propagation of uncertainty, if only as an approximation in case of a non-Gaussian distribution of $f_i(t)$. In principle, γ_{eff} can also be produced from series of PL to overlapping intervals $[t_i, \infty)$, which avoids bias due to the choice of t_n and t_m. Here, a point-wise error could only be obtained through the fit-error. To distinguish between this definition and the usual effective exponents, these will be called *tail effective exponents* in the following.

The asymptotic exponent can then be read off as

$$\gamma = \gamma_{\text{eff}}(t \to \infty) \quad .$$

The functional form of $\gamma_{\text{eff}}(t)$ is related to the form of $f(t)$ and needs to be approximated to obtain a meaningful extrapolation for $t \to \infty$. Assuming PL corrections, the first order exponent can be determined from a direct PL fit or by rescaling the abscissa to linearize the tail of $\gamma_{\text{eff}}(t)$.

The hard part is now to find a proper estimate of the error $\Delta\gamma = \Delta\gamma_{\text{eff}}(t \to \infty)$. The least squares fit performed to obtain the extrapolated value provides a measure of statistical uncertainty for the fit-parameters, which include the asymptotic value of γ. However, this statistical error is only meaningful if the fitted form actually is a valid hypothesis describing the data. The suitability of the form may be quantified by the sum of residuals. However, since the actual functional form of $f(t)$ is often not known, there is an infinite space of functions which would have to be tested. Fitting a PL to the tail can only be viewed as a first approximation. Even then, chosen form may often not fit to the data for all t but only above some cutoff $t > t^{\text{min}}$. This produces two more sources for uncertainty:

First, the choice of t^{min} influences the optimal fit parameters, normally outside the bounds given by the statistical fit error. This suggests, that the statistical error of the fit parameters is basically meaningless. Formally, that error would decrease and ultimately vanish for $t^{\text{min}} \to \infty$, but the relative noise $\Delta f(t)/\langle f(t)\rangle$ increases with t. This is especially problematic for observables like autocorrelation or -response functions, which decrease with time such that $\Delta f(t) \gtrsim \langle f(t)\rangle$ after a time t^*. Effective exponents calculated according to equation (2.4) become meaningless beyond t^*, thus another $t^{\text{max}} < t^*$ must be chosen, which again influences the fit result.

Secondly, the choice of the model function may be a good first approximation, but without representatively sampling the whole available function space one cannot quantify how much the result will change when adding a second or third order. In many cases it would not even be computationally feasible to add higher orders, comprising more fit parameters, because of a very limited number of available data points. For example, in the case of a finite size scaling analysis, the extrapolation is not done for $t \to \infty$ but for $L \to \infty$ where all previous arguments hold, replacing t by the system size L.

These sources for uncertainty are more significant than the statistical error of the single fit and are hard to be quantified in a well-defined manner. In some studies one and the same observable is obtained from simulations with different values of a parameter. Often, the asymptotic time evolution of this observable does not depend on this parameter, based on physical arguments, of which universality is an example. If this is the case the different exponents obtained for each value of the parameter can be averaged and the standard deviation or the spread of the values can be used as an estimate of the error $\Delta\gamma$. The standard error is not used, because the different estimates are not independent Gaussian random variables.

If in such a case the form of corrections can also be assumed to be independent of the parameter, the corrections can be better approximated by consistency arguments: Multiple forms may be used to extrapolate the asymptotic value of a universal exponent, then the form most likely describing the corrections best is the one which minimizes the spread of the values obtained for different values of the parameter. This is attempted for the scaling exponent β in the $N > 1$–RSOS model in section 4.1.2.

These difficulties result from the fact, that the described computer experiments are more of an exploratory nature when it comes to the finite-time regime. With no hypothesis about the form of $f(t)$ available for testing, the numerical points of $\langle f(t) \rangle$ provide the sole basis from which to derive the asymptotic, $t \to \infty$, behavior. Thus, the quality of the desired results is limited by the amount of numerical information available about $\langle f(t) \rangle$, any attempts of extracting features or extrapolating to regimes not covered by the simulation results in increased uncertainty. This is the main reason why simulations need to be extensive: First, the numerical form of $\langle f(t) \rangle$ must be known with little statistical error, requiring large sample sizes N, where large systems can reduce the number of required samples through self-averaging. Second, the asymptotic regime must be approached as close as possible. In cases where t is the simulation time in the scaling regime, this does not only necessitate long simulations, but also simulations of very large systems to avoid artifacts from a potential crossover into a steady state.

2.2. From Continuum- to Atomistic Models

Very large scale systems can be treated using methods from continuum mechanics, where separate particles are not considered, only phases spread out continuously in space. These systems formally contain numbers of particles which are well within the thermodynamic limit. Examples for continuum models can be found for fluid-dynamics in the Navier–Stokes equations or the Burgers equation [28], in models of phase separation, like precipitation [13, 29] or spinodal decomposition [30, 31] as well as models for surface growth, like the Kardar–Parisi–Zhang (KPZ) equation [32], the molecular beam epitaxy (MBE) equation [7, 33], and interface instabilities, like the Kuramoto–Shivashinsky (KS) equation [34]. Where no analytical solutions are available, such equation can be integrated numerically. However, when studying the formation and evolution of nanostructures in solids, and especially on lattices, structures are of interest which are to small for the *continuum limit* to be considered a good approximation. The problem of nucleation is the most prominent example of this: A cluster forms when single particles nucleate, which cannot be described in the continuum limit.

Atomistic simulation by solving the equations of motion for many particles (MD) quickly becomes infeasible when the total number of particles in the system increases. To bridge this gap, MC methods were introduced [3, 35]. They are designed to enable the investigation of larger systems and longer times than can be achieved using other atomistic simulations using limited compute power. MC methods are based on simple rules designed to reproduce the statistical properties of the physical system that is to be modelled. Discrete MC models can be computationally much less expensive than directly solving stochastic partial differential equations numerically.

Physical systems are described by a given Hamiltonian which in the absence of external driving forces is of the form:

$$H(\mathbf{q}) = T(\mathbf{q}) + V(\mathbf{q}) \quad ,$$

where $\mathbf{q}$ is a $(d \cdot N)$-vector of all particle coordinates in a d–dimensional system with N particles. $T(\mathbf{q})$ and $V(\mathbf{q})$ describe the kinetic energy and the potential energy of the system, respectively. The kinetics of the particle movement can be obtained by solving the dynamical equation, which can be written as Hamilton's equations:

$$\frac{\partial p_i}{\partial t} = -\frac{\partial H}{\partial q_i} \quad \text{and} \quad \frac{\partial q_i}{\partial t} = \frac{\partial H}{\partial p_i} \quad ,$$

where $\mathbf{p}$ is $(d \cdot N)$-vector of the momentum components of all particles. The state of the system can be described by a vector in phase-space $(q_1, \ldots, q_{(d \cdot N)}, p_1, \ldots, p_{(d \cdot N)})$. The equations can be integrated numerically, which is done in MD.

Systems with many particles can be described more efficiently using statistical mechanics and thermodynamics. With sufficiently many particles, the state-space of the system can be reduced to its configuration space $(q_1, \ldots, q_{(d \cdot N)})$. The probability for the system to be in an arbitrary state q is given by the Boltzmann-distribution:

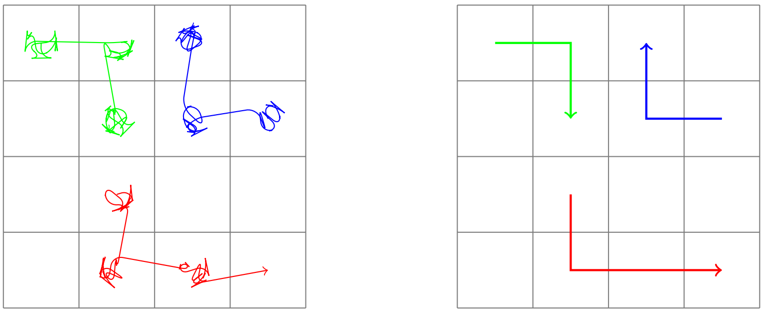
$$P(\mathbf{q}) = \frac{1}{Z} \exp\left(-\frac{E(\mathbf{q})}{k_B T}\right) \quad , \tag{2.5}$$

where Z is the partition sum of the system, $E(\mathbf{q}) = V(\mathbf{q})$ is internal energy of the system in state $\mathbf{q}$, k_B is the Boltzmann constant and T denotes the temperature.

To study the evolution of large bulk systems driven by thermal fluctuations, the Metropolis algorithm [36] was devised which is produces a Markov chain of states following the distribution (2.5). The algorithm works by starting at a current state $\mathbf{q}^j$ and proposing a randomly chosen movement of a particle, resulting a new state $\mathbf{q}^{j+1}$. This update is accepted with transition probability depending on the difference in internal energy between the states $\Delta E = E(\mathbf{q}^{j+1}) - E(\mathbf{q}^j)$. The basic idea is illustrated in figure 2.1. Classically, the Metropolis transition probability is used:

$$P(\mathbf{q}^j \to \mathbf{q}^{j+1}) = \begin{cases} \Gamma_0 \Gamma(\mathbf{q}^j, \mathbf{q}^{j+1}) & E(\mathbf{q}^{j+1}) < E(\mathbf{q}^j) \\ \Gamma_0 \Gamma(\mathbf{q}^j, \mathbf{q}^{j+1}) \exp\left(-\frac{\Delta E}{k_B T}\right) & E(\mathbf{q}^{j+1}) > E(\mathbf{q}^j) \end{cases} \quad , \tag{2.6}$$

where Γ_0 is the rate at which updates are performed. In simulations, Γ_0 is usually equal to one. In an experimental system it would be on the order of the frequency



(a) Molecular Dynamics (b) Lattice Monte-Carlo

Figure 2.1.: Diffusion trajectories of three particles on a square lattice as seen in MD and lattice Monte-Carlo simulation using the Metropolis algorithm, left and right panel, respectively: (a) MD models thermal fluctuations (ideally) deterministically. Particles spend most of their time oscillating at lattice sites, jumps between sites are rare because they require diffusing particles to obtain high kinetic energy from collision with matrix particles (not shown, but part of the simulation). (b) In MC thermal particle movement is not explicitly simulated, but modeled by the Boltzmann distribution of states (2.5). Only jumps between sites are attempted in the simulation and accepted with finite probability, according to equation (2.6).

of thermal oscillations (Debye frequency). The rate

$$\Gamma(\mathbf{q}^j, \mathbf{q}^{j+1}) = \exp\left(-\frac{E_{\text{barrier}}(\mathbf{q}^j, \mathbf{q}^{j+1})}{k_B T}\right) \tag{2.7}$$

can be added to model diffusion barriers. Other transition probabilities are possible, such as Glauber dynamics [37].

2.3. Models for Phase Ordering and Nanostructure Evolution

One of the best known example of models the Metropolis algorithm can be applied to is the Ising model [38] of interacting magnetic moments. When only nearest neighbor (NN) interactions are taken into account, the Ising system without any external magnetic field is described by the Hamilton:

$$H = -J'_{\text{NN}} \sum_{\langle ij \rangle} s_i s_j \quad , \tag{2.8}$$

where J'_{NN} is the coupling strength between neighboring parallel magnetic moments and $\langle \cdot, \cdot \rangle$ denotes a sum over NN pairs. In this model competitive growth is domains in a ferro, or anti-ferromagnet, by changing the sign in equation (2.8), can be studied.

2.3.1. The Kinetic Metropolis Lattice Monte-Carlo Method

While in the original Ising model all proposed updates are spin-flips (non-conservative), a conservative version was introduced in which spins are exchanged (Kawasaki dynamics). [39] In this picture, the two spin-orientations $s_i = \pm\frac{1}{2}$ can also be viewed as particles of two different species $c_i \in 0, 1$, allowing the study of processes of phase separation and self-organization in bulk-systems. This method is called 3D kinetic Metropolis lattice Monte Carlo (KLMC) [6, 18, 19, 40] and can be generalized to include more than two species, by providing a matrix of binding energies between particles two species m and n: J^{mn}.

2.3.2. The Potts Model

A special case of the matrix of binding energies is the q-states Potts model [41, 42]:

$$J^{mn} = \delta_{mn} \quad , \tag{2.9}$$

for $0 \leq m, n < q$. For $q = 2$, this model is equivalent to the Ising model with the transformation $s_i = 2c_i - 1$, which gives $J^{\text{Potts}} = 4J^{\text{Ising}}$. The Potts model can be mapped to problems like grain growth in crystals [43].

The Ising model shows a continuous, second order transition to an ordered phase below a critical temperature T_c in its critical dimension $d = 2$ and above. In the Potts model this phase transition is of first order, dis-continuous, above a critical number of states q_c, for example $q_c^{d=2} > 4$ [44] and $q_c^{d=3} = 3$ [45]. In two dimensions, the critical temperature Potts models are analytically known [41, 46]:

$$K_c = \ln(1 + \sqrt{q}) \quad \text{with} \quad K = \frac{J}{k_B T} \quad , \tag{2.10}$$

where K denotes the dimensionless effective temperature.

In the ordered phase, for all pure Potts models, domain growth can be observed which follows different growth laws depending on whether the employed dynamics is conservative or not. In non-conservative Potts models domain growth follows the Lifshitz–Cahn–Allen law, exhibiting a PL with a growth exponent $\phi = 1/2$ [47, 48], while for spinodal decomposition in systems with conserved order parameter the consensus in literature is, that the Lifshitz–Slyozov law with a scaling exponent $\phi = 1/3$ [13] is followed asymptotically [49, 50]. When concentrations are conserved with one being sufficiently dominant, precipitation and Ostwald ripening can occur [6, 12, 13, 29].

Another field of study is the domain growth in the presence of quenched disorder [46]. In such systems, the potential term in the Hamiltonian has site-dependent contributions, where the site dependence is random but fixed for each realization of an evolving system. Possible types of disorder are, among others: Random field disorder, where a site-dependent external field is applied:

$$H = -J_{\mathrm{NN}} \sum_{\langle ij \rangle} c_i c_j + B_i \cdot c_i$$

Another type is random bond disorder, where the couplings between sites vary:

$$H = -\sum_{\langle ij \rangle} J_{ij} \cdot c_i c_j \quad ,$$

here without external field. A commonly investigated version of the latter uses a bimodal distribution of disordered bonds, where $J_{ij} \in J^0, J^1$. This also covers the case of broken bonds or bond-dilution, where $J^0 > 0$ and $J^1 = 0$, or vice versa. In this case, the critical temperature is shifted. For small fractions of broken d, the new critical temperature can be approximated using a mean-field approach, assuming the missing bonds lead to reduced effective coupling between spins. This yields the shifted critical temperature:

$$K_{c,\mathrm{MF}} = \frac{K_c}{1-d} \quad , \tag{2.11}$$

where K_c is the critical temperature of the pure systems, given in equation (2.10).

The presence of disorder can smoothen first order phase transitions to second order transitions [51] and slows the growth of domains. An open question for many systems is whether the type growth law is changed in the presence of disorder, or if the growth law is super universal [52–54]. In the case of valid super-universality (SU), the form of the growth-law present in a pure system would not change in the presence of disorder, as long as the disorder does not affect the final state of the system, e. g. a ferromagnetic system remains ferromagnetic even in the presence of such disorder. With a growth following a PL, only the exponent may depend on the disorder parameters.

Some studies found SU growth laws in disordered magnets [55–57]. In random bond and random field Ising models a crossover to a late time regime with logarithmic scaling was observed, violating SU [52, 58].

Spin glasses, a type of disordered spin systems are also of special interest because they are the type of problem the quantum annealer device D-Wave is built for [59,60]. A better numerical understanding of these systems can help to pose hard problems to such a device and test its quantum properties.

To precisely investigate growth laws in a system, the growth needs to be followed over several orders of magnitude, which requires simulations of large systems. Especially to distinguish between slow PL growth and logarithmic growth, a system must be followed over long times. In the case of disordered systems, required to average over many disorder realizations (independent runs) in addition. Examples for the applicability of the present work are given in section 5.1, where computational performance is being analyzed in section 3.4.1.

2.4. The Kardar–Parisi–Zhang and Edwards–Wilkinson Universality Classes

The Kardar–Parisi–Zhang (KPZ) equation [32] describes an accepted standard model for the growth of surfaces under random deposition of particles:

$$\partial_t h(\mathbf{x},t) = v + \underbrace{\nu\nabla^2 h(\mathbf{x},t)}_{\text{surface tension}} + \underbrace{\lambda[\nabla h(\mathbf{x},t)]^2}_{\text{loc. growth vel.}} + \eta(\mathbf{x},t) \tag{2.12}$$

Here, v is the average growth velocity of the surface, which is usually eliminated by a transformation into the co-moving frame: $h + vt \to z$. The second term models smoothening of the surface at finite temperatures in a diffusional manner. The strength of this surface tension is set by the parameter ν. The third term introduces a spatially varying local growth velocity with an amplitude λ. It is motivated by the dependence of the height-change induced by the deposition of single particles on the local slope of the surface.

Randomness in the influx of particles is modeled by a noise term $\eta(\mathbf{x},t)$. The noise is usually considered as a zero-average and Gaussian, which is how it turns out naturally in MC simulations. The variance

$$\left\langle \eta(\mathbf{x},t)\eta(\mathbf{x}',t')\right\rangle = 2D\delta^d(\mathbf{x}-\mathbf{x}')(t-t') \tag{2.13}$$

requiring it to be uncorrelated in both space $\mathbf{x}$ and time t. The variance of the noise $\eta(\mathbf{x},t)$ is defined only to an amplitude D.

Equation (2.12) in general describes the growth of a d–dimensional surface into dimension $d+1$. In $d=1$, it resembles the stochastic Burgers equation [28] and as such represents a generalization to arbitrary dimension. This connection links the Kardar–Parisi–Zhang (KPZ) model to fluid dynamics and the problem of randomly stirred fluids [61], where it would describe a velocity profile instead of a surface height profile. There are further examples of systems which evolve following this equation. Among them are, in $d = 1$, the propagation of flame fronts [62, 63], the surface morphology of growing cancer cells [64] and even magnetic flux lines in superconductors [65]. In $d = 2$, a further example is directed growth of polymers in random media [66, 67]. Aspects of all these systems belong to the KPZ universality class, which is defined by equation (2.12). Figure 2.2 shows a two–dimensional example of a surface structure created by KPZ growth.

An interesting property of a growing surface or interface is its roughness, which is defined as:

$$W(L,t) = \sqrt{\langle h^2(\mathbf{x},t)\rangle_{\mathbf{x}} - \langle h(\mathbf{x},t)\rangle_{\mathbf{x}}^2} \quad , \tag{2.14}$$

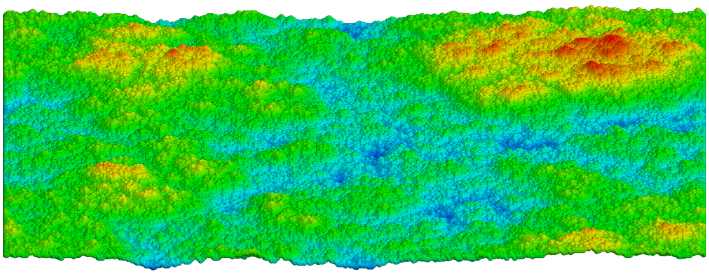


Figure 2.2.: Illustration of a two–dimensional surface embedded in three–dimensional space. The colors indicate the height of each point, with red peaks and blue valleys. The depicted instance represents a KPZ surface in the steady state.

where $\langle\ \rangle_{\mathbf{x}}$ denotes an average over all spatial coordinates. This growth process is expected to follow a PL described by the scale-invariant Family-Vicsek scaling law [68]:

$$W(L,t) \sim L^{\alpha} f(t/L^{z}) \quad , \tag{2.15}$$

where the scaling law $f(u)$,

$$f(u) \sim \begin{cases} u^{\beta} & \text{for } u \ll 1 \\ \text{const.} & \text{for } u \gg 1 \end{cases} \tag{2.16}$$

is universal for the KPZ class. Here, α is the roughness exponent, describing the stationary state, where the correlation length exceeds the lateral system size L. The growth regime is governed by the growth exponent β. The ratio of these gives the dynamical exponent $z = \alpha/\beta$.

In the case $\lambda = 0$, equation (2.12) turns into a stochastic diffusion equation, called EW equation [69]. This case shows different universal behavior with $f(u) \sim \log u$ for $u \ll 1$ and a dynamical exponent $z = 2$. Since solutions to this equation are known analytically, this class can serve to test implementations of models, where the parameter λ can be adjusted accordingly.

Due to the stochastic nature of this problem, all observables are determined as an example average over multiple independent runs. In numerical studies, large systems show ideal self-averaging. This means, increasing the volume of the simulated system by a factor m reduces the sample variance by m which, at fixed sample size n, reduces the standard error by $\sqrt{m}$. This is the same effect as increasing the sample size a factor m and comes at about the same computational cost.

Two different surface growth models will be considered here: The octahedron model [70] and the restricted solid-on-solid model (RSOS) [71], also known as Kim–Kosterlitz model. Both models can be generalized for arbitrary dimensions, while here the focus lies on $2+1$ dimensions, which is is the most relevant case in technical applications.

2.4.0.1. Physical Aging

Because a growing surface with the roughness scaling introduced above evolves over time, properties of the system change, which does not happen in equilibrium or in a steady state. The process of a physical system changing during its evolution away from some initial condition, here a flat surface, is called physical aging. It was first observed in glassy systems, but similar effects can also be observed in other systems, like magnets evolving towards equilibrium and non-equilibrium systems [72]. Physical aging is commonly investigated by analysing the two-time autocorrelation $C(t,s)$ and autoresponse $R(t,s)$ functions, where aging can be observed in the *aging-regime*, at times sufficiently long after the waiting time s [72,73]: $t-s \gg t_{\mathrm{micro}}$, where t_{micro} denotes a system dependent microscopic time scale. These observables will be introduced in sections 4.2, equation (4.18), and 4.3, equation (4.24), respectively.

In surface growth simulations, aging can only be observed in the growth regime, where dynamical scaling is taking place. In order to study aging of a growing surface over long times, it is necessary to consider large systems, so that $u \ll 1$ in equation (2.16) for all times of interest, i. e. to stay away from the steady state. For this reason, the studies in this work would not be possible without efficient parallel implementations of the considered models.

2.4.1. The Octahedron Model

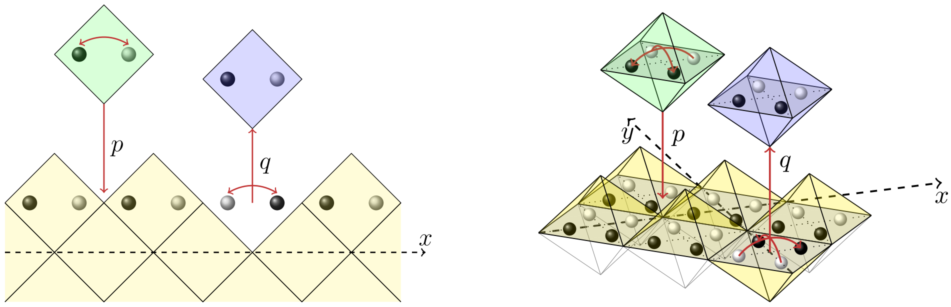
The octahedron model [70, 74] is a generalization of the roof-top model [75, 76] for $1+1$–dimensional surface growth to higher dimensions. These models restrict the height differences (slopes) between neighboring lattice sites to $\sigma_{x/y} = \pm 1$. In $1+1$ dimensions this can be illustrated (fig. 2.3a) as stacking squares, with one corner pointing up, where for each square the lower two edges must be supported by upper edges of squares in the layer below (direct stacking of corner on corner is forbidden). In the generalization to higher dimensions, hyperoctahedra are stacked the same way. In [70] it was shown, that the octahedron model in higher dimension does indeed exhibit KPZ scaling.

The height at each site is measured at the upper corner of the hyperoctahedron occupying it. No two neighboring sites can have the same height, since a height difference of $\sigma_{x/y} = 0$ is not allowed. With this restriction, the smoothest possible surface takes a zig-zag shape and thus exhibits a roughness of $W_{\mathrm{flat}} = 0.25$.

Figure 2.3b illustrates the octahedron model in $2+1$ dimensions, where deposition or removal of particles on the surface, follow the generalized Kawasaki rules [70, 74]:

$$\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} \underset{q}{\overset{p}{\rightleftharpoons}} \begin{pmatrix} 1 & -1 \\ 1 & -1 \end{pmatrix} \quad , \tag{2.17}$$

where allowed deposition processes are carried out with probability p and removals with probability q. In this representation of the update rules, each line corresponds to one spatial direction. Thus the rules are in general comprised of d lines for a $d+1$–dimensional surface. In the one–dimensional case, these rules belong to



(a) $1 + 1$d roof-top model (b) $2 + 1$d octahedron model

Figure 2.3.: (a) Illustration of the lattice gas dynamics (white and black balls) in the $1 + 1$–dimensional roof-top model and the mapping to deposition (p) and removal (q) of squares at appropriate lattice sites. (b) Illustration of p and q processes in the $2 + 1$–dimensional octahedron model. The octahedron surface can be viewed like an outer product of two roof-top surfaces (roof-top surface $\otimes$ roof-top surface).

an asymmetric exclusion process (ASEP) [75] in a lattice gas. In d dimensions, they correspond to an ASEP of d-mers, moving along the bisectrix of the Cartesian coordinate axes.

Commonly, the case of only depositions taking place is investigated ($q = 0$ and $0 < p \leq 1$, pure KPZ). This corresponds to a totally asymmetric exclusion process (TASEP), where all particles, or d-mers, are moving in only one direction. The case $p = q$, where effectively no growth is taking place, shows scaling in the EW universality class. The EW fixed point is unstable in $d = 1$ [26], that is, only points on the line $p = q$ exhibit EW scaling. For any $q \neq p$, the surface will be attracted by the KPZ ficed point and roughen asymptotically. The EW fixed point is stable in $d > 2$ and there is a roughening transition at some value $|p - q|_c > 0$. $d = 2$, the dimension treated in this work, is the lower critical dimension for this transition [26]. Using modified, longer range, update rules the model can be extended to other types of surface growth, like MBE and KS [77].

In simulations, the noise depends very sensitively on the randomness of the updates. Especially the pure KPZ case is very sensitive to correlations introduced by the random updates. Thus, this model also presents very sensitive test for the freedom from correlations in the employed parallel implementation. Mainly autocorrelation function in the $2 + 1$–dimensional dimer-TASEP turn out to be a good indicator. Hence, a detailed study can provide valuable insight also for parallel implementations of other models which are less sensitive, where correlation may have more subtle effects.

The model is set in the co-moving frame of the surface. To describe the morphology of the surface in this reference frame, only the slopes between sites are required, which can only take two different values each ($\sigma_{x/y} = \pm 1$). Each slope can be encoded in a single bit, thus for each lattice site only two bits of information need to be stored.

2.4.2. The Restricted Solid on Solid Model

The RSOS model, or Kim–Kosterlitz model, [71] implements the growth process in any dimension by stacking of hypercubes. Here, the surface height at each site is equal to the number of cubes stacked there. Depositions and removals are only allowed as long as the absolute height difference between neighbors stays below a parameter N and are then carried out with probabilities p and q, respectively. It falls into the same universality classes as the octahedron model, as λ in equation (2.12) can be adjusted through p and q in the same way.

Even though this model does allow NN site of equal height and thus supports actual flat initial conditions ($W(L, t = 0) = 0$), the growth process still shows an intrinsic width, due to discretization noise caused by discrete height differences. [78]

Simulations of the RSOS model are performed in the co-moving frame of the surface, thus only requires encoding of the slopes between sites. Most commonly, simulations of this model are performed for $N = 1$ [71, 79, 80] but studies with $1 < N \lesssim 10$ exist [81, 82] and recently [83]. The case $N > 1$ for this model is sometimes referred to as generalized RSOS. For $N \to \infty$, the model crosses over into a random deposition model, where the scaling exponent becomes $\beta_{\text{rnd. dep.}} = 1/2$ [81].

3. Parallel Implementation: Towards Large-Scale Simulations

All models described in the previous chapter have in common, that they are coarse-grained in time yet atomistic. Built from simple rules, they are very efficient in sequential computation. Still, a need for even more efficient computation and utilization of more compute power arises, when the problem size, in terms of system size or time scales of interest (usually both), becomes large enough. For example, in the case of 3D kinetic Metropolis lattice Monte Carlo (KLMC) this is the case when pushing the limits of atomistic simulations to experimental scales which are normally only within the range of continuum methods. For surface growth models, ever larger systems and longer times are required to increase the accuracy of estimated values for some universal growth and aging exponents.

Ways to speed up the Metropolis algorithm have been implemented for a long time, often to speed up relaxation in the Ising model. Examples are cluster algorithms like the Swendsen-Wang [84] or the Wolff algorithm [85], which allow flipping whole clusters of spins, thereby circumventing energy barriers that occur when the cluster has to be flipped spin-by-spin. Another way are iterative sequential updates [86], where the lattice sites to be updated are selected in order, which speeds up the propagation of changes in the system by introducing a preferred direction. The latter approach breaks the condition of detailed balance, but still satisfies the weaker balance condition. Both approaches have in common, that they make the system evolve to the correct equilibrium configuration faster, by changing the kinetics. Such approaches will not be discussed further, since this work focusses on kinetic models.

Another frequently used approach in Ising model simulations is updating based on a checkerboard decomposition of the lattice, herein called stochastic cellular automaton (SCA) [87] dynamics. This technique may also have a detrimental effect on kinetics, but, depending on parameters, to a lesser extent than the aforementioned algorithms for fast relaxation. The influence of SCA dynamics on autocorrelation functions in the octahedron model will be discussed in section 4.2.3. However, by avoiding random memory accesses, an SCA allows implementations to access memory more efficiently and can be parallelized or vectorized in a straight-forward manner. Ever since GPUs have been introduced this has been exploited successfully [88–91] to speed up simulations.

For kinetic models, ideally, random-sequential (RS) dynamics[1] should be used, where single updates are statistically independent. Only this can ensure that diffusion is correct (unbiased) and updates do not introduce spatio-temporal correlations. Since single updates are not computationally expensive, only performing very many of them is, such an algorithm cannot be straightforwardly parallelized. A solution to this problem is to apply domain decomposition (DD) in such a way that random updates can be performed simultaneously in non-interacting domains and resulting correlations are suppressed. This has been done for the KLMC method on GPUs using Compute unified device architecture (CUDA) as subject of my diploma work [93].

---

[1] The term “random sequential” is adopted from a study of update procedures for the 1d asymmetric exclusion process. [92]

For completeness, an alternative method for kinetic Monte Carlo simulations shall be mentioned: At low temperatures, or in systems with few frequently moving particles, the Metropolis algorithm will reject most, even almost all, of the suggested updates. In such situations, rejection-free algorithms are more efficient. The earliest example of this class of algorithms is the Bortz–Kalos–Lebowitz (BKL) algorithm, or n-fold way. [94] Here, instead of choosing a random update and deciding whether to accept or reject it, an update is chosen according to the probability of it being accepted in the Metropolis algorithm and is then carried out with probability one. Since this requires all possible updates to be evaluated and sorted, single updates are much more computationally expensive than in the Metropolis algorithm, which is thus more efficient in situations with many probable updates available.

Rejection-free algorithms can also be parallelized using domain decomposition (DD), but parallel implementations tend to scale poorly due to the necessity of keeping updates globally sorted and load-balancing issues. [95, 96] They can be implemented in a scalable way by implementing them less rigorously or when the computations required to evaluate updates are sufficiently expensive. [97] A brief comparison of the performance of a parallel Metropolis implementation and a sequential implementation of the waiting time method (WTM) [98] is presented at the end of section 3.4.2. In principle, it can be assumed that for sufficiently low temperatures and late stages, for example in coarsening simulations, even a single CPU rejection-free implementation may be faster than any parallel implementation of KLMC.

With respect to parallel compute efficiency, continuum methods have the distinct advantage that they naturally expose a sort of data parallelism: For one time-step the changes at each point of the system is computed only from a stencil of points taken from the previous time-step, independently of the changes for all other points. This, in principle, enables using more compute resources in parallel when the system size is increased. For MC methods with RS updates, introducing parallel workloads using DD can incur correlations and change results, which is a core problem attacked in this work. The aforementioned SCA approach can be implemented in parallel exactly and efficiently, and it is thus worthwhile considering it for applications where complete absence of correlations is not required.

In this chapter, first important features of selected current parallel architectures are introduced in section 3.1, accompanied by a short reiteration of efficient techniques for encoding of on–lattice data. In section 3.2, a variety of DD schemes for RS dynamics are described, where, on the implementation side, the focus is put on the $(2 + 1)$d octahedron model. Two GPU implementations of SCA dynamics for the octahedron model are described in section 3.3. Finally, section 3.4 introduces an alternative method to utilize the fine-grained parallelism of GPUs for stochastic lattice models.

3.1. Parallel Architectures and Programming Models

The sequential compute power of CPUs has almost not been increased for years. This is because established ways to increase it have become less feasible to implement. The most straightforward way to increase single-core performance is to increase the clock frequency, which increases the rate at which instructions are executed. This comes with increased power consumption and thus heat which needs to dissipated, both introduce limiting factors. Other approaches are branch prediction, out-of-order execution and pipelining, which mostly serve to hide latencies for memory accesses. Pipelining increases the number of instructions which are executed per clock cycle by executing multiple instructions partially in parallel by overlapping them, creating an execution pipeline. These approaches come with a cost in additional transistors required to implement the logic responsible for optimizing the instruction flow, which again cost energy.

The latter approach already points towards parallelization as the only feasible way to increase the compute power available for a calculation. In principle, combining multiple identical compute units in one machine is quite efficient, since the gain in compute power scales linearly with the number of compute units and thus energy consumption. Actually using the combined compute power theoretically made available this way thus poses a separate problem and the possible parallel speedup s is limited by the amount of parallel workload r_{par} in the algorithm which is to be implemented in relation to the sequential portion r_{seq} (Amdahl's law) [99]:

$$s = \frac{1}{r_{\text{seq}} + \frac{r_{\text{par}}}{n}} \quad , \tag{3.1}$$

where n is the number of processor cores used. The parallel portion r_{par} can sometimes be increased by changing the algorithm, which, even if it reduces the performance of the program on a single core, can lead to higher performance in massively parallel setup with large n.

Details of the parallel programming required also depend on the architecture at hand. In the following, aspects of CPU, GPU and heterogeneous architectures are described, which are relevant to the MC application considered in this work. The range of CPUs and GPUs which were used in benchmarks, presented later, are listed in table 3.1. The numbers of cores per device give an impression of the parallelism required, but are not directly comparable across architectures in terms of compute power. The theoretical peak performance of a compute device is usually measured in floating-point operations per second (FLOPS). This figure is omitted from the table, because floating point operations only make up a small amount of the computational load of the MC codes presented here. The number of update attempts per second is

Table 3.1.: Properties of compute devices used for benchmarks. The numbers of cores given in parenthesis correspond to the total number of virtual cores available through hyper threading (see section 3.1.1). Numbers given for memory bandwidth (Bw.) are theoretical peek performances. The thermal design power (TDP) can provide a rough estimate of the maximum power consumption under load. The Tesla K80 is a dual-GPU board, of which a single GPU code will only use one GPU.

Device	Cores	Mem. Bw. [GB/s]	TDP [W]	Release
Xeon X5650	6(12)	32	95	Q1'2010
i7-4930K	6(12)	59.7	130	Q3'2013
Tesla C2070	448	144	247	Q3'2010
Tesla K80 (1 GPU)	2496	240	150	Q3'2014
GTX Titan Black	2880	336	250	Q1'2014
GTX Titan X	3072	336.5	250	Q1'2015

a more relevant performance measure for the applications presented here. The table lists the peak memory bandwidth between main memory and CPU or between global device memory and GPU, respectively. These peak values are often not achieved in benchmarks because of throttling under heavy load to limit power consumption and heat production. This power envelope is given by the TDP of a device, which is the maximum sustained amount of heat produced by a device and is thus also related to the power consumption. The actual power consumption can deviate and depends on computational load and other factors.

3.1.1. CPU

Cluster compute nodes usually contain two or four CPUs, which, today, in turn contain between six and 24 cores each. Thus, current compute nodes consist of twelve to 96 CPU cores, sharing a common main memory. The fact that the memory can be accessed by all cores can be best taken advantage of by programs running multiple threads (multi-threading, shared-memory parallel applications), which execute parallel tasks relying on common data or on only a logical decomposition of a total shared dataset. This type of parallelism is straightforward to use, employing, for example the built-in threading support in C++11 or OpenMP.

In order to reduce latencies when accessing frequently used data, CPUs[2] provide multiple levels of caches. When a datum in main memory is accessed, the CPU will always load the whole *cache line*[3] containing the requested datum, thus if another datum in the same memory region is accessed later, it can be loaded directly from the cache with no need to wait for the main memory, while the corresponding cache line remains resident. This means, that memory accesses are more efficient, if successive addresses are accessed or if random accesses are restricted to a small region in the address space.

---

[2] Only x86 CPUs have been used in this work, but most statements in this section are most likely also true for other architectures like Power.

[3] The sizes of cache lines vary and can even be different for different cache levels in the same CPU. On an i7-4930K, the cache-line size is 64 B for all levels.

Each core contains a small amount of level-1 (L1) cache of usually 64 kB or 128 kB on Intel or AMD CPUs, respectively. Often, half of this is reserved for code, half for data. Additional levels 2 and 3 are shared between multiple cores, thus the total size of these caches is shared between threads in a multi threaded program or between simultaneously running processes. Furthermore, modern CPUs are *cache coherent*, which means, that if a chunk of memory (cache line) is resident in the cache of multiple cores, the contents are guaranteed to agree among all caches of all cores. Enforcing this can lead to severe slowdowns if one core is writing to an address falling into a cache line which is resident in the cache of another core, for which it must be invalidated and read again from main memory when required. For example, this situation would arise when using internal states of random number generators (RNGs) for different threads which are stored in a single array.

Some CPUs offer *hyper threading*, which means, that each core contains more than one set on registers to hold the state of multiple tasks (i. e. threads). This allows the CPU to quickly switch to another task if the first one is stalling, for example when it is waiting for a memory transaction to complete. Thus, running more threads than there are physical cores, up to the hyper threading capability, helps to hide latencies.

Another type of parallelism is vectorization, realized in CPUs by providing vector registers which can hold vectors of scalar values (32- or 64-bit integers or floating point values, or others) and corresponding vector instructions which perform an operation on all scalar elements of these vector simultaneously. This pattern is called single instruction multiple data (SIMD). Modern compilers try to vectorize code when instructed to do so, but this only works well with loops the iteration length of which is known at compile time.

3.1.2. GPU

In a way, GPUs rely on the SIMD pattern in a more flexible form: A GPU contains a number of vector processors, containing many simple cores, or processing elements[4] but only one instruction unit, so that all cores can only execute the same instruction in parallel. Contrary to SIMD, the control flows of the these processing elements are allowed to diverge, which leads to sequential execution of different instructions assigned to the processing elements. This pattern is thus called single instruction multiple thread (SIMT). A group of 32 SIMT threads thus locked is called a *warp*. Apart from the allowed control flow divergence among threads, the main difference between these concepts is, that each thread can in principle perform random accesses to memory and has its own independent set of registers, while in SIMD data is handled in fixed vectors (SIMD words), which are loaded in to vector registers crossing SIMD elements.

---

[4] ... or CUDA cores (NVIDIA) or Stream Processors (AMD).

It depends on the approach chosen for a specific program, whether (a) SIMT threads are used more like actual threads or (b) basically like SIMD elements, with a warp taking the role of a SIMD unit:

(a) In a MC simulation, where sites, jump-directions and types of updates are chosen randomly or based on a local state, the thread picture is useful. Each thread may perform updates partially independently, but as many operations as possible, such as random number generation, should be implemented collectively to avoid unnecessary warp-divergence.

(b) The SIMD picture is a natural fit for operations, operating on vectors of data, such as integrations or linear algebra operations. However, cellular automata and SCA versions of MC algorithms do fit this pattern as well.

Each vector processor contains a small amount of fast memory used both as *shared memory* and L1 cache[5] This is the only memory region on a GPU where random accesses are efficient. Shared memory can be use to cache data loaded by one thread which will be required by another thread later. MC algorithms with random site selection may be implemented in such a way, that a chunk of the simulated system, which can be entirely held in shared memory, is updated collectively by all threads in a vector processor (section 3.2). More vectorized code may use shared memory to exchange data between threads (section 3.3).

The GPU does also provide a larger amount of global memory, which can be accessed by all vector processors, which currently amounts to up to 12 GB on NVIDIA GPUs. The bandwidth of global memory is much larger than that of the main memory for the CPU (see table 3.1), it is still smaller when put in relation to the compute performance of the device. Accesses to global memory work by loading larger chunks of data than single words, akin to the cache lines on CPU. Ideally, each warp triggers one transaction of $32 \cdot 4$ bytes at consecutive addresses.[6] If threads access global memory randomly, separate transactions of this size are performed for each thread separately, which constitutes very inefficient use of global memory bandwidth. In order to allow non-linear an non-coalesced access patterns to global memory, additional caches exist: A level-2 cache and the read-only caches for texture and constants. The latter are commonly used for lookup tables.

Accesses to global memory can be ordered by loading consecutive chunks of data into shared memory, using it as a sketchpad, as mentioned above. Simpler efficient access patterns for global memory are streaming (section 3.3) or warp-collective (random) coalesced accesses (section 3.4).

As long as there are sufficient resources, it is possible to run many times as many threads on a GPU than there are cores. This is also necessary in order to hide latencies and achieve high compute throughput. For example, a GTX Titan Black (NVIDIA Kepler generation) GPU features 15 vector processors, each supporting up to 2048 threads, yielding a maximum and when achievable, ideal, total number of 30720 resident threads on the device.

---

[5] On current NVIDIA GPUs (since Fermi): 64 kB of memory, of which 48 kB are addressable shared memory and 16 kB L1 cache, or vice-versa, depending on the programmer's choice.

[6] The lowest address should ideally be located at a 128 byte boundary (aligned access).

Threads on the GPU are grouped into *threads blocks*, each of which is executed by a one vector processor. Each vector processor can have multiple resident blocks, as long as sufficient resources in terms of shared memory, registers and scheduling capability are available.

Programming Model To a program running on a computer (host), a GPU presents itself as a compute accelerator to which specific tasks (*kernels*) can be offloaded. The main program is being executed on the CPU. Data required for GPU computations is transfered to the GPU global memory, before it can be used by a kernel on the device. Data can be transfered from main memory, or directly from a different GPU on newer devices, which can be useful in multi-GPU applications.

Data transfer from main memory via PCI Express is rather slow (below 8 to 16 GB/s), but if lots of transfers are required during a computation, it is possible to perform transfers while a kernel is being executed on the device, which does not require the transfer to be complete. Thus time required to transfer data can be overlapped with computation.

3.1.3. Heterogeneous Parallelism and MPI

Basically any compute node containing GPUs provides more than one CPU core per GPU. In some cases it can be beneficial to utilize the parallel compute resources on both GPU and CPU. An example is performing on-the fly analysis on the CPU while the GPU continues to work on a simulation, as discussed in section 3.5.

Another level comes into play when compute resources installed in multiple physical nodes are to be used. Separate compute nodes share neither the operating system nor memory, thus different instances of a program need to run on each node. These processes can exchange data by sending messages over a network. The standard protocol used for such communication is the Message Passing Interface (MPI). The requirement to send data over a network between nodes adds the problem of communication latencies, which is not present in shared-memory parallel programs, but is similar to the problem of transferring data via PCI Express between a GPU and main memory.

When one large dataset is to be processed, one way to overlap communication with computation can be to split the data set into chunks. After transferring the first chunk, computation on the first chunk can be started, while the next chunk is being transferred.

3.1.4. Bit-Coding of Lattice Sites

A common property of many stochastic lattice models is, that each lattice site can only take a small number of different states, thus only little information needs to be stored per site. The smallest addressable unit of data is one byte, containing eight bits, which is also the amount of information it can hold. Modern architectures provide registers holding words of four (32-bit, GPU) or eight (64-bit, CPU) bytes.[7] Storing only one or two bits of information, corresponding to two to four states in any of these units is not an efficient use of resources.

Let b be the number of bits required to encode a lattice site. Then, given a word size w in bits, each word can be interpreted as a linear memory region in which w/b lattice sites can be stored. Usual d–dimensional addressing can be used to encode chunks of a d–dimensional system, which are as close to a (hyper-)cube as possible. For example: In $d = 2$, with $b = 2$ and $w = 32$ (GPU) squares of 4×4 lattice sites can be encoded. This allows keeping random memory accesses as local as possible and caching chunks with as little surface as possible in small memory regions, like GPU shared memory.

Details of this type of encoding are presented in appendix A.1. This encoding is called *synchronous* [100], because it encodes lattice sites of the same system in one word. Another form of synchronous bit-coding is discussed in section 3.3.2. It is based on the idea of implementing SCA updates in a bit-vectorized fashion which is often used in MC simulations of the Ising model [35, 101] and has also been successfully applied to the RSOS model (multi-lattice-site coding in [79]).

A different approach to bit-coding, where sites of different realizations, then simulated parallel, are encoded in one word [102], may be called *asynchronous* [100]. A very recent study of bit-vectorized Ising spin-glass simulations on GPU using this approach can be found in [100]. Here, asynchronous bit-coding is not employed, but an application of the general idea to SIMD processing on GPU in section 3.4.

3.2. Domain Decomposition for Stochastic Lattice Models

Direct parallel implementation of stochastic lattice models like surface growth models or Metropolis MC methods with random-sequential (RS) dynamics is not possible. In order to generate a Markov chain of states incremental updates are performed, which by themselves do not contain independent computations which could be executed in parallel. This is holds true even for rejection-free methods.

The alternative is to generate and apply many update attempts concurrently, which creates parallel workload that can be distributed over workers like CPU cores, i. e. threads or MPI ranks, or multiprocessors or scalar cores of a GPU. To generate a Markov chain in parallel requires these update attempts to be statistically independent in the sense that an integrated transition over a time Δt of parallel updates follows the same probability distribution as the integrated transition over Δt of sequential updates. This implies, that a balance condition which is fulfilled by the sequential algorithm must also be fulfilled by the parallel version. Speaking of an integrated transition allows for relaxing on detailed balance.

---

[7] SIMD words are not considered here, since these are vectors of 32-bit or 64-bit scalars. This means, for example, that a shift operation is applied to each scalar word separately.

This can in theory be done in a rigorous way (Shim and Amar [96] did this for the n-fold way algorithm), but will then require an overwhelming amount of communication, and especially synchronization of workers, and will thus result in poor parallel scaling. A more practical approach is to use domain decomposition (DD) to generate domains where workers can perform updates needing to communicate. Update attempts in these non-interacting domains will be performed asynchronously for a time t_{async} which will usually be one Monte Carlo step (MCS), that is one sweep of the domains (*full update*). For the time t_{async} each domain constitutes a smaller system with fixed boundary conditions (FBC), thus this is only an approximation to the pure sequential algorithm. The goal in choosing a DD scheme is to minimize effects of the fixed boundaries on the evolution of the system. For a fixed scheme, reducing t_{async} does systematically reduce the effects of this approximation.

3.2.1. DD for Asynchronous Updates

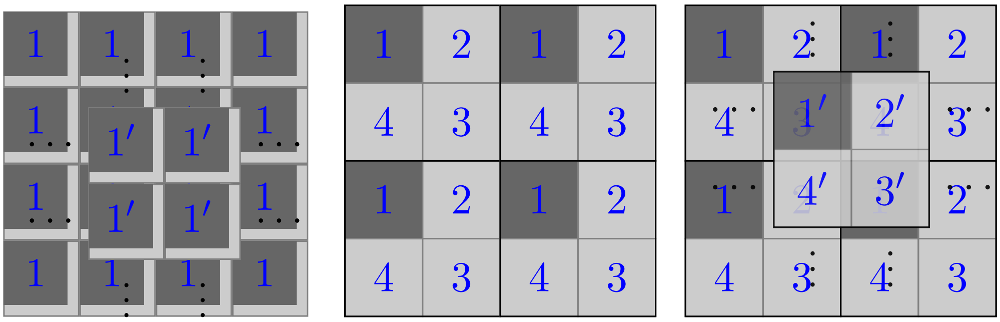
DD schemes shall first be treated generically. When the given problem consists in distributing simulation lattice among multiple *workers* which are to perform full updates asynchronously, the solutions can be formulated independent of the parallel architecture. In a specific implementation, the role of workers may be taken by threads or MPI ranks in a multi-CPU setup or thread blocks on GPUs.

3.2.1.1. Dead border (DB)

In the dead border (DB) DD scheme, used by the GPU implementation of the octahedron model in [27, 103], decomposes the system into tiles, where the rim of each is kept inactive during asynchronous updates (*dead border*), so that no update of the site in the active part of a tile would affect or be affected by the state of site in a neighboring tile. If a border site is chosen for an update attempt the update is not carried out. After the asynchronous interval t_{async}, the origin of the tiling is moved randomly before the next asynchronous updates, displacing the dead borders so that former inactive border sites may be updated and propagate changes between formerly separate tiles. Since not all sites are active during a sweep of the lattice, a time step under DB is a little less than a complete sweep:

$$1\text{MCS}_{DB} = 1/(N_{\text{latticesites}} - N_{\text{bordersites}})\text{MCS} \tag{3.2}$$

In general, such as in KLMC, where each update attempt requires interaction between the updated site and all its NNs, sites on the rim of tiles in all directions need to belong to the dead border. In the octahedron model on a square lattice, where only the links between sites carry state (the slope), only one rim in each



(a) dead border (DB) (b) double tiling (DT) (c) DT DD with random origin (DTr)

Figure 3.1.: 2d schematics of DD schemas that have been evaluated for parallel implementations of the $(2+1)$d octahedron model. All schemes can be straightforwardly applied in setups of arbitrary dimension. Dark areas indicate regions being updated concurrently while the light areas are inactive, acting as buffers. Numbers indicate a (randomized) sequence of synchronous steps in which the asynchronous domain updates are performed. Domains labeled with primed numbers illustrate a possible randomized decomposition after moving the decomposition origin: $O \to O'$. Domains calculated based on a random origin will always wrap around the system (periodic boundary conditions (PBC)), even if PBC are not used in the simulation. DB ((a)) is displayed for the special case of the octahedron model (and other slope-based surface growth models) where updates can affect neighboring cells only in positive x or y direction. In general, all cell edges would be dead borders.

spatial direction needs to be inactive. This is because, for each direction one slope is encoded on-site (to the left neighbor), only the other one must be retrieved from the right neighbor. Thus updating a site on the left rim of a tile will not require access to the neighboring tile to the left. This also holds for all slope-based surface growth models on rectangular lattices, including RSOS. Figure 3.1a illustrates DB DD in 2d.

In the implementation of the octahedron model, each 32-bit word encodes 4×4 lattice sites, thus, freely picking a random origin would result in words becoming shared between neighboring tiles. For reasons of performance, the early implementation thus restricted movement of the DD origin in such a way, that the borders would always be located at the edge of 32-bit words. Essentially the origin was moved randomly on a coarse grid with steps of 4×4 lattice sites. This variation will henceforth be labeled coarse dead border (cDB). DB was also implemented without restriction of the DD origin to a coarse grid, by increasing the width of the dead boarder to five lattice sites. This padding ensures that if a word is shared between tiles, the left tile will not need to write to it, because all lattice sites encoded in this word, belonging to the left tile, will be inactive and will not have an active neighbor. Properties of these two variations are discussed in section 4.2.1.

3.2.1.2. Double tiling (DT)

Parallel implementations of KLMC use double tiling (DT) for DD, this scheme is illustrated in 2d in figure 3.1b. Here, the system is decomposed into tiles, which are split into two sub-tiles in each spatial direction, creating 2^d sets of non-interacting domains, where d is the dimension. Theses domain sets are updated in a random order at each MCS, synchronization occurs after completing each sweep of a domain-set. Sub-tiles do not comprise inactive sites, thus updates of lattice sites on the rim of a sub-tile will affect sites in neighboring sub-tiles which are at that time inactive.

In DT borders of DD domains always remain at the same place, which enables higher performance, because, not having to deal with arbitrary decomposition origins, memory alignment can be controlled and the amount of data that needs to be exchanged between workers (specifically MPI ranks) is reduced. The disadvantage is, that it allows errors of the effective FBC approximation to accumulate at the locations of the domain boundaries. Nevertheless, it has turned out to give results of sufficient quality for KLMC [93]. Essentially this scheme was also used for a less rigorous implementation of the n-fold way algorithm [95].

3.2.1.3. DT DD with random origin (DTr)

In a model where the only source of randomness is the random selection of lattice sites, it must not be biased. Thus, a DD scheme which allows imbalances in the site-selection to accumulate at specific places (sub-tile boundaries), however slightly, cannot be expected to perform well in general. The accumulation can be removed by randomly moving the decomposition origin like with DB (see figure 3.1c). In the DT DD with random origin (DTr) scheme restricting the random origin to a coarse grid is not necessary in any case, since there effectively are dead borders of the same width as the decomposition domain, which easily provides enough padding to avoid having words shared between workers. An analysis in section 4.2.1 shows this DD scheme to be free of correlations to statistical accuracy, this is why it was used in most of the octahedron-model simulations presented in chapter 4.

3.2.1.4. Implementation

Shared memory implementations, such as multi-threaded CPU code, of these schemes are rather trivial, since the DD can be implemented logically: It suffices to restrict the random site selection of each worker to an appropriate region, without physically moving or copying data. CPU performance will profit if the decomposition domain easily fits into the L1 cache. Thus, it can be advantageous to decompose the system into a number of smaller domains, which may exceed the number of worker threads available.

Using the same approach on a GPU, i. e. storing the system in global device memory and letting threads take the role of workers, would be very inefficient: Such an implementation would utilize the available memory bandwidth very inefficiently due to the non-coalesced memory accesses and would suffer from large latencies. A alternative approach to essentially this is described in section 3.4. For the current purpose thread blocks need to be regarded as workers at this level (*device-layer*).

In GPU code sweeps of domain sets are performed inside a GPU kernel. The subsequent synchronization takes place when the kernel terminates. To perform a sweep of an assigned decomposition domain, a thread block loads it into the shared memory and writes the data back to global memory only after completing the sweep (see next section). The size of device-layer domains is set to be as large as the amount of available shared memory per multiprocessor permits, in order to maximize the number of threads that can be run per block. Another conceivable option would be to use smaller domains and fit multiple blocks onto a multiprocessor to reach the same utilization (maximum number of threads per multiprocessor). A block would only need to encompass one or two warps. This alternative would make less efficient use of the very limited amount of shared memory, because more memory would be used by domain borders and thus inactive lattice sites. Using larger domains also reduces the negative effect of the FBC approximation implied at this level of DD.

In order to use multiple multi-CPU nodes or GPUs, data needs to be exchanged between these units, which may well be equated with MPI ranks. Any of the above implementations and DDs may be used, where each rank is only assigned a subset of all decomposition domains which cover slice of the whole system. Usually, it is most efficient to distribute ranks in only one spatial dimension, because this minimizes the number of neighboring ranks each ranks need to exchange data with. The edges of the ranks' meta-domains have to be exchanged at each synchronization point. In case of DT this means communicating the borders and in cases with random DD origin (DB and DTr) this means communicating the overlap areas induced by the transition from origin O to O'. As part of the present work, this has been implemented for KLMC [104], but it will not be elaborated upon.

3.2.2. Second DD Layer on GPUs

On GPUs, thread blocks assume the role of workers in the aforementioned DD schemes (*device-layer*). Thus a device-layer domain needs to be updated in parallel by a usually large number of threads, where each thread performs updates on a, consequently small, block-layer domain. In principle, all DD schemes discussed before (see also figure 3.1), can be applied here, with only one difference: Since block-layer domains are inevitably small (often only 4×4 lattice sites), errors introduced by the FBC approximation would be large when performing a full sweep. Instead, only *single-hit* updates are performed by threads, which consist in performing only one update attempt before synchronization. In this scheme, time does effectively not progress asynchronously in independent domains ($t_{\text{async}} \sim$ single update attempt ~ 0), since no update depends on a previous update which was not synchronized with the other workers.

3.2.2.1. Single-Hit DT

Using single-hit DT, each thread is logically assigned a tile of 2^d block-layer domains, which is called the *thread cell (TC)*. The smallest choice is $2 \times 2 (\times 1)$ words[8] This configuration is written using $\log_2$ as `TC=1,1`. This gives a smallest block-layer domain of one word or 4×4 lattice sites in the 2d octahedron model. A variant with random origin was also evaluated.

3.2.2.2. Single-Hit DB

In case of single-hit DB the TC and block-level domain are identical, which allows for smaller TCs. The smallest possible configuration would be denoted `TC=0,0`, containing only one word. Because of the small block-level domains, making the fraction of border sites rather large, and the lack of asynchronism, single-hit DB is implemented comprising *delayed* borders rather than dead ones. Thus, if an update attempt hits the border of a block-level domain, the thread will wait until the bulk updates are completed and evaluate the update attempt afterwards. Updates hitting the corners of domains are applied after that. In this way the number of dead border sites in equation (3.2) is zero thus sweeps are complete.

Since the decomposition origin is moved without restriction, words are likely to be shared between neighboring TCs. Atomic operations are therefore used to update the domain in shared memory.

For the octahedron model, in cases where either $p > 0$ or $q > 0$, which is the usual case when simulating the KPZ universality class, updates can never be allowed for two NN sites at the same time, because slopes are restricted to ± 1. Thus, in this case, delayed borders are not required to avoid conflicts between updates and updates ignoring borders are completely equivalent to updates with delayed borders. Hence, in the present work, all simulations stated as using DB at block-level are actually ignoring borders at block-level if only one of p and q is finite.

3.2.2.3. DD Parameters for the Octahedron Model

Properties of all these methods are analysed in sections 4.1.1 and 4.2.1. The size and shape of thread cells is given in the notation `TC=`$\log_2(x)$`,`$\log_2(y)$, where x and y are the numbers of words in each direction.

A device-layer domain is updated by a number of *threads* (T) in a configuration `T=`$\log_2(t_x)$`,`$\log_2(t_y)$. Since each thread is assigned to one thread cell, the size of a device-layer domain is $2^{\mathtt{TC+T}}$ words or $2^{\mathtt{TC+T+2,2}}$ lattice sites.[9] This notation could be extended to any number of dimensions, including 3d KLMC, but it is only used in the context of the 2d octahedron model here.

---

[8] 32-bit words, containing 4×4 (2d octahedron model) or $4 \times 4 \times 2$ (KLMC) lattice sites.

[9] The expression `x,y` can be read both as a linear dimension $\log_2 \vec{l} = (x, y)$ and as a volume $\log_2 V = x + y$, depending on what is of interest to the reader. Define $\log_2$ of a vector element-wise.

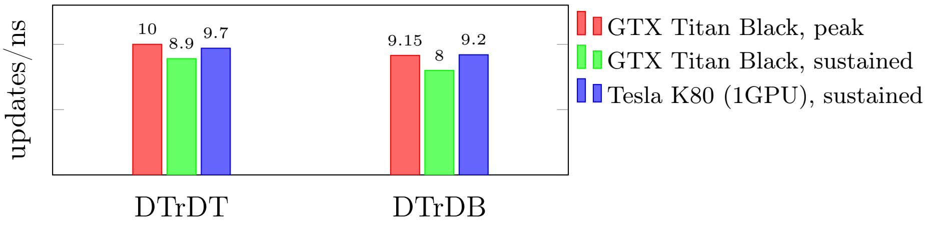


Figure 3.2.: Performance of the DTr at device level and single-hit DT at block level (DTrDT) and DTr at device level and single-hit DB at block level (DTrDB) variations of the RS implementation of the octahedron model on GPU. For K80, the sustained performance equals the peak performance. Benchmarks were performed for systems of lateral size 2^{16}.

3.2.3. Performance

The implementations for future calculations use DT DD with random origin (DTr) at device-layer and differ in the type of block-layer DD employed: The two relevant variations are DTr at device level and single-hit DT at block level (DTrDT) and DTr at device level and single-hit DB at block level (DTrDB). The performance of these is presented in figure 3.2. On a GTX Titan Black GPU, the performance drops after the first $\sim 100\,\mathrm{MCS}$, because the device clocks down under load. This leads to a measured sustained performance which is lower than the peak. The performance on a Tesla K80 is about constant. The DTrDB variant is consistently slower, by about ten percent.

The sequential implementation on an i7-4930k CPU delivers 0.055 update attempts per ns for a system of lateral size $L = 2^{12}$, a size where the while system fits into the L3 cache. The performance is less for larger systems, which do not fit into L3 cache. Running multiple independent runs on the same device is also likely to reduce performance. A parallel implementation, using DTr, running twelve threads, performs 0.28 update attempts per ns. This leads to a speedup factor of about 30 for the GPU code over a single socked CPU.

3.3. Lattice Level DD: Stochastic Cellular Automaton

Parts of this section, including sub-sections 3.3.2 and 3.3.3 have been published as a conference proceedings paper [105].

The checkerboard–SCA approach achieves a sweep of the lattice (MCS) in a well-defined and efficient manner by first updating the even lattice sites, then the odd ones. The sets are defined by the parity of the exclusive or of the coordinates $(x \oplus y) \wedge 1$, analogously to the black and white squares of a checkerboard. Since this eliminates the process of random site-selection, the updates can be ordered in a way which results in most efficient, usually linear, memory access patterns.

3.3.1. Local Approach for the Octahedron Model

The straightforward way of encoding tiles of 4 × 4 lattice sites in 32-bit words, as described in section 3.1.4 has the advantage that NN sites, which are required to perform an update may be found in the same word, making it the encoding of choice if lattice sites are selected randomly. Based on this encoding, an SCA implementation has been created which takes advantage of the allowed linear memory access pattern.

In the GPU implementation, four kernel calls are quired per MCS: For each combination of odd/even lattice sites and odd and even chunks of lines. The system is split into chunks along the y direction to avoid write-write conflicts between blocks. A block is used like and extended SIMD unit, with each thread processing a scalar element. The system is processed line by line, with the current and the next line being cached in shared memory. Threads iterate over all active sites in their respective 32-bit words to perform updates. The remaining details are similar to the implementation described in the next section.

This approach has the advantage of using the same lattice encoding as the RS implementation, which makes this a good prototype for an SCA implementation because it can be use in conjunction with analysis codes already in use with the RS implementation.

3.3.2. Non-Local Approach for the Octahedron Model

For bit-vectorization, the ideal encoding is non-local with respect to the relations of data in configuration space. Instead the bits are grouped in memory according to their role. The SCA defines four roles in total, based on two binary properties: The direction of the slope $\sigma_{x/y}$ and the parity of the lattice site (odd/even). This grouping is illustrated in the right-hand panel of figure 3.3: All slopes with the same role are stored at consecutive addresses in memory, allowing vectorization with any word size not larger than half the size of the simulation cell $X/2$.

3.3.2.1. Bit-Vectorized GPU Implementation

The kernel of the bit-vectorized GPU implementation is summarized in pseudo-code in figure 3.4. Care must be taken regarding the parity of rows with respect to the considered sub-lattice (line 3.4.2): All slopes belonging to sites in odd rows are perfectly aligned and no shifting is required. In even rows the σ_{x^+} slopes, stored at the NN site in x direction, are shifted by one bit in memory (compare figure 3.3, left).

The implementation treats consecutive words processed by threads of the same block as one effective SIMD word, with a maximum effective size of $w_{\text{eff,max}} = w \times \mathit{maxThreadsPerBlock}$, where w is the word size in bits, which is 32 on GPU, giving $w_{\text{eff,max}} = 2^{15}$. This SIMD-word size can be adjusted to span the simulation cell, as long as the lateral size does not exceed $X = 2^{16}$. Larger simulations are rarely required, thus the kernel displayed in figure 3.4 assumes the effective SIMD word to span the system. The work assigned to thread blocks is distributed along the

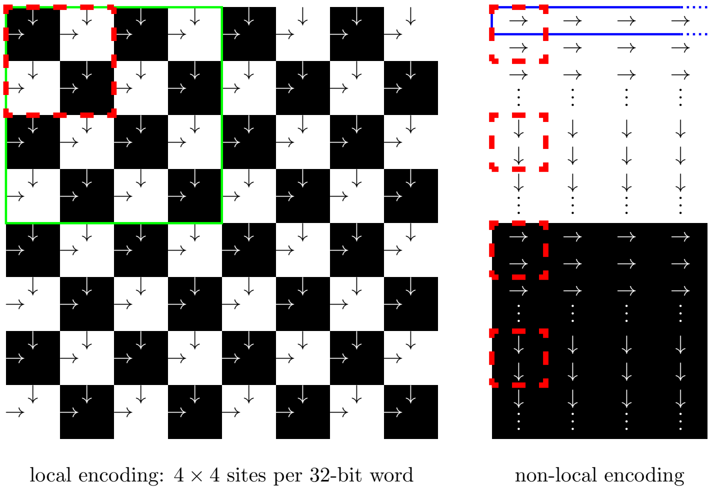


Figure 3.3.: Comparison of direct, local bit-coding (left) and non-local encoding suitable for vectorization. Black and white areas represent even and odd lattice sites, respectively. Arrows represent slopes connecting neighbors in the indicated direction. Red dashed frames show the correspondence between locally and non-locally encoded slope information. Solid frames indicate data stored in a 32-bit word.

y direction. Buffer regions between blocks or global atomics are not employed since, as long as all blocks update the same sublattice, the non-locality of the encoding of on-site slopes avoids write conflicts in global memory.

Each thread updates 32 lattice sites simultaneously. The corresponding slopes $\vec{\sigma}_{x+}$ are placed in a buffer in shared memory ($\vec{\sigma}_{x+}^{\text{shared}}$) to facilitate block-wide rotation when updating even rows. The correct set of $\vec{\sigma}_{x+}$ slopes is compiled in the lines following 3.4.8. An xor-mask m encoding the Kawasaki exchanges to be carried out is used to apply updates in parallel. Again, $\vec{\sigma}_{x+}$ needs to be treated separately in even rows: The update mask is applied in parts to the corresponding data in shared memory. Atomics could be used to apply m, in order to remove the need for synchronization in line 3.4.21, but this was found to yield lower performance.

The update mask m in line 3.4.14 is generated according to the Kawasaki rules in equation (2.17), which can be implemented by the following relations:

$$m_p = \vec{\xi}_p \wedge \neg(\vec{\sigma}_{x-} \vee \vec{\sigma}_{y-}) \wedge \vec{\sigma}_{x+} \wedge \vec{\sigma}_{y+} \tag{3.3}$$

$$m_q = \vec{\xi}_q \wedge \neg(\vec{\sigma}_{x+} \vee \vec{\sigma}_{y+}) \wedge \vec{\sigma}_{x-} \wedge \vec{\sigma}_{y-} \tag{3.4}$$

$$m = m_p \oplus m_q \quad , \tag{3.5}$$

where $\vec{\xi}_r$ denotes a word of random bits set with probability r. If $q = 0$, the calculation of m_q can be omitted and $m = m_p$.

Generation of $\vec{\xi}_{0.5}$ is trivial and fastest, provided a good pseudo-random number

```
Require: $X \times Y$                                          ▷ system size
Require: $w$                                                   ▷ word size in bits
Require: $\vec{\sigma}$                                        ▷ bit-vector of slopes, one word
Require: $\vec{\sigma}^{0/1}_{x/y^-}[X/w/2, Y]$                ▷ 4 arrays of slopes ($x/y$, even/odd)

for $y \in blockBounds$ do
    $par_{\mathrm{row}} \leftarrow par_{\mathrm{lat}} \otimes \textsc{parity}(y)$
    $x_w \leftarrow t_{id}$                                    ▷ index of slope-vector in $x$

    $\vec{\sigma}_{x/y^-} \leftarrow \vec{\sigma}^{par_{\mathrm{lat}}}_{x/y^-}[x_w, y]$
    $\vec{\sigma}_{y^+} \leftarrow \vec{\sigma}^{\neg par_{\mathrm{lat}}}_{y^-}[x_w, y+1]$      ▷ NN $y$
    $\vec{\sigma}_{x^+} \leftarrow \vec{\sigma}^{\neg par_{\mathrm{lat}}}_{x^-}[x_w, y]$        ▷ NN $x$
    $\vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}] \leftarrow \vec{\sigma}_{x^+}$                  ▷ shared buffer for rotation

    if $\neg par_{\mathrm{row}}$ then                          ▷ rotate SIMD word by one bit:
        $\textsc{synchronizeThreads}$
        $\vec{\sigma}_{x^+} \leftarrow \textsc{shiftRight}(\vec{\sigma}_{x^+}, 1)$
        $\vec{\sigma}_{x^+} \leftarrow \vec{\sigma}_{x^+} \vee \textsc{shiftLeft}(\vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}+1], w-1)$
    end if

                                                               ▷ compute xor mask for update:
    $m \leftarrow \textsc{updateMask}(\vec{\sigma}_{x^-}, \vec{\sigma}_{y^-}, \vec{\sigma}_{x^+}, \vec{\sigma}_{y^+}, p, q)$

                                                               ▷ apply mask:
    $\vec{\sigma}^{par_{\mathrm{lat}}}_{x/y^-}[x_w, y] \leftarrow m \oplus \vec{\sigma}^{par_{\mathrm{lat}}}_{x/y^-}[x_w, y]$
    $\vec{\sigma}^{\neg par_{\mathrm{lat}}}_{y^-}[x_w, y+1] \leftarrow m \oplus \vec{\sigma}^{\neg par_{\mathrm{lat}}}_{y^-}[x_w, y+1]$

    if $\neg par_{\mathrm{row}}$ then
                                                               ▷ apply mask to NN $x$ slopes, except LSB:
        $\vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}] \leftarrow \vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}] \oplus \textsc{shiftLeft}(m, 1)$
        $\textsc{synchronizeThreads}$
                                                               ▷ apply mask to LSB of next NN $x$ slopes:
        $\vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}+1] \leftarrow \vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}] \oplus \textsc{shftRight}(m, w-1)$
        $\textsc{synchronizeThreads}$
        $\vec{\sigma}^{\neg par_{\mathrm{lat}}}_{x^+}[x_w, y] \leftarrow \vec{\sigma}^{\mathrm{shrd}}_{x^+}[t_{id}]$
    else
        $\vec{\sigma}^{\neg par_{\mathrm{lat}}}_{x^+}[x_w, y] \leftarrow m \oplus \vec{\sigma}_{x^+}$
    end if

end for
```

Figure 3.4.: Bit-vectorized GPU kernel using non-local encoding to update one sublattice with given parity ($par_{\mathrm{lat}} = (x \oplus y) \wedge 1$). It is executed for each sublattice in order to complete one MCS. PBC apply for all coordinates, including indexes in the shared memory buffer $\vec{\sigma}^{\mathrm{shrd}}_{x^+}[]$. t_{id} is a short-hand for the thread ID within the thread block. The above directly applies if the thread block ($\hat{=}$ SIMD word) spans the system in x direction. Statements involving $\vec{\sigma}_{x/y^-}$ represent two separate operations on $\vec{\sigma}_{x^-}$ and $\vec{\sigma}_{y^-}$. Load and store operations for parameters and states of random number generators happen before respectively after the presented loop and are omitted for brevity. See text for details.

generator with all bits following a uniform distribution is employed. For an arbitrary r, it is necessary to generate $\vec{\xi}_r$ sequentially using w random numbers. All $w = 32$ random bits need to be generated. Generating only those random bits which are required for the actually possible updates in a way which minimizes warp-divergence, does not improve performance. Even then the implementation is considerably faster than a non-vectorized version using local encoding.

With the effective SIMD word spanning the simulation cell in one direction, the implementation utilizes global memory perfectly efficiently: All accesses are coalesced and all bits read are required to perform updates. To handle even larger systems ($X \geq 2^{17}$), the kernel needs to be changed. The word index x_w, initialized in line 3.4.3, is iterated with a stride equaling the number of threads per block. The procedure remains unchanged for odd rows, where no slopes need to be shifted. For even rows, the shared buffer $\vec{\sigma}_{x^+}^{\text{shared}}$ holds two SIMD words. $\vec{\sigma}_{x^+}$ slopes need to be shifted between neighboring SIMD words, thus the whole of a second SIMD word is cached to ensure keeping global memory accesses coalesced. The buffer is used in a wrap-around fashion ($\vec{\sigma}_{x^+}^{\text{shared}}[x_w \mod bufferSize]$) and the first word $\vec{\sigma}_{x^+}[0]$ is cached in shared memory separately in order to apply PBCs without having to read the data from global memory twice.

Good quality pseudo-random numbers are crucial in this implementation, especially in the optimized case $r = 0.5$. While in some cases adequate for Monte-Carlo methods, a linear congruential generator (LCG) is insufficient here: Correlations would severely influence results and produce accumulating errors since the SCA updating procedure is itself correlated and is to be decorrelated by the random acceptance of updates in the first place. Also, LCGs do usually not provide good randomness of single bits. The TinyMT [106], used in this work (see section A.3) yields good results in the present implementation.

For performance comparisons, a multi-threaded CPU implementation using a word size $w = 64$ bit was created for the case $p = 0.5, q = 0$. The usage of SIMD instructions through the vector extensions of GCC [107] did not increase performance. This may be because some operations, like bit shifts across a whole SIMD word, require more operations than with scalar commands, which compensates the performance gain due to vectorization.

3.3.3. Performance of SCA Implementations

Tests of the CPU implementation were performed by running the maximum number of hardware threads provided by the hyper-threading capabilities of the platform (12). This increases the performance by less than one percent over running only one thread per physical core (6) for the bit-vectorized implementation. This suggests that the performance of the bit-vectorized implementation on CPU is not significantly limited by memory latency. The gain is about 20% for the local implementation.

The achieved performance of different implementations is presented in figure 3.5 as the number of update attempts performed per nanosecond. In the course of one MCS, each slope needs to be touched twice, translating into two write and two read

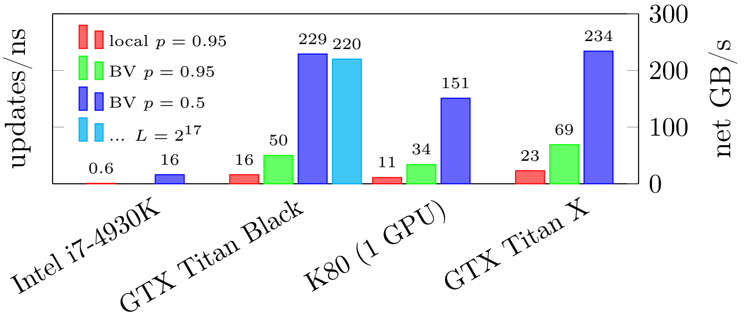


Figure 3.5.: Benchmark results of various SCA implementations of the octahedron model on Kepler (GTX Titan Black, K80) and Maxwell (GTX Titan X) generation NVIDIA GPUs and on an Intel i7-4930K CPU. The benchmarks on GPUs were performed for systems of lateral size $L = X = Y = 2^{16}$ (except for the cyan bar, where $L = 2^{17}$, see text). Benchmarks on CPU used a lateral size of $L = 2^{14}$ lattice sites. All presented benchmarks were performed for $q = 0$, p as given in the legend. The right axis translates the rate of performed updates into the net bandwidth required to read and write the processed data, excluding any overhead.

accesses to each slope per MCS in the ideal case. This is illustrated by the right axis of the plot associating the performance with the ideally required bandwidth. For $p = 0.5, q = 0$, the bit-vectorized implementation performs about 229 updates per nanosecond on a GTX Titan Black GPU, which requires 229 GB/s of slope data to be transferred between global device memory and the GPU. The NVIDIA Profiler `nvprof` reports an achieved memory bandwidth of about the same value, since the memory transfers required to load and store states and parameters of random number generators negligible. This bandwidth is lower than the maximum bandwidth listed for the device in table 3.1, but it is equal to the bandwidth reported by the `bandwithTest` utility from the CUDA SDK for pure device-to-device memory copy. At the same time `nvprof` reports low to medium utilization of arithmetic units. Thus the code is clearly memory-bound.

When the lateral system size is increased to $X = 2^{17}$, the effective SIMD word does no longer span the simulation cell and memory accesses cannot be done as efficiently anymore. However, the resulting drop in performance remains below 5% (cyan bar).

In the case of arbitrary p, the generation of correctly distributed random bits becomes more expensive. The benchmarks for $p = 0.95$ (green bars in figure 3.5) show a performance reduced by a factor of four to five, which is still faster than the local implementation for the same case (red bars). Since memory access patterns of the bit-vectorized implementation remain unchanged between the cases $p = 0.50$ and arbitrary p, the decrease in performance can only stem from the increased computational load of random number generation, which does not require branches. This means, that the implementation turns from memory-bandwidth-bound to compute bound. Specific choices for the probabilities, which are the sums or differences of fractions $1/2^i$ can be generated at lower computational cost than arbitrary ones, since they can be generated from less than 32 random numbers using the logical "and" and "or" operations.

The newer GTX Titan X (Maxwell-generation) GPU provides a significant speedup over Titan Black for arbitrary p, due to increased compute power, but only a marginal gain in the case $p = 0.5$, since the available memory bandwidth is almost the same.

The values listed for TDP in table 3.1 may not be perfectly comparable across vendors, but can provide a rough estimate of the power consumption of the device in a steady state under load. The actual power consumption during the benchmarking was not measured. Comparing performance and TDP for the Kepler-generation cards, one can conclude that the lower clocked, compute GPU K80 is more energy efficient running the presented implementations than the gaming card.

The case $p = q = 0.5$ was also tested, but not separately optimized, because simulations of this case are of less scientific interest since it falls into the analytically-solved Edwards–Wilkinson universality class. However, evaluating additional $q = 0.5$ updates does not significantly affect performance since memory bandwidth remains the main limiting factor.

3.4. The Multi-Surface Coding Approach

The chief disadvantages of the two-layered DD approach described in section 3.2.2 originate from the need to cache chunks of the simulated system in a small amount of fast, local memory, to avoid random accesses to slow global memory. This works, as long as the memory required per lattice site does not exceed about two bits. The need to cache is removed in the SCA approach by throwing out random site-selection, among other things, allowing to store more information per lattice site. Neither of these approaches will work when significantly more than one bit of information is encoded per lattice site and random site-selection is desired. Implementing the RSOS model with $N > 1$ or the Potts model with random-bond disorder or $q > 4$ cannot be done using the two-layered DD approach because any reasonably sized domain would take up too much memory to be cached in local memory. There may be no efficient way to solve this problem on GPU in general.

These two approaches described above focus on simulating large systems over long time scales. They are less efficient at simulating small systems, since the parallel workload created is not enough to completely occupy a GPU below a certain system size, which depends on the algorithm and the device. When small systems are of interest, usually a large number of samples is required. This is the case, for example, when performing finite-size scaling, since the region of interest is reached sooner and one can scale over a larger range when starting at a smaller system size. Averaging over many small systems is also preferable over few large runs in cases where self-averaging is not present or weak. Large systems do no provide self-averaging over disorder realizations and self-averaging was found to be weak for response functions in the Ising Model [108].

The multi-surface coding (MS) approach takes advantage of the fact that multiple realizations of the same simulation are to be run and vectorizes over realizations instead of parallelizing the simulation. This is usually done for models with a small number of states (few bits per lattice site) and thus employing bit-vectorization as described in section 3.3.2. For example, it has been done for $N = 1$ RSOS [109] and previously for the 3d Ising model [110] on 64-bit CPUs, but the technique traces back to Ito and Kanada [102]. It has also been used for the exchange MC method, or parallel annealing, relevant for spin glasses. [111] In the present context, however, this method will only serve as a role model: Bit-vectorization is only appropriate when the number of possible states a lattice site can take is small ($\lesssim 4$) while here the goal is to efficiently handle at least one byte of information per lattice site.

Since caching of domains in local memory is not possible under these conditions, random accesses to global memory must be made efficient. All threads of a warp should access data within the same 128 byte region of global memory, ideally at a 128 byte offset (coalescing, see section 3.1). The MS approach allows treating these 128 byte chunks as effective SIMD-words: With a warp size of 32 threads, the SIMD-word is a vector of 32 four-byte scalars, where each scalar value could encode information from one system (layer) of a stack of 32 systems (multi-surface). The properties of the scalars will be discussed later, with the specific applications in mind.

3.4.0.1. Vectorization

Vectorizing MC updates over realizations is done by selecting a shared random coordinate for the whole warp. Information belonging to this coordinate in all layers of the multi-surface is stored in the same vector, which can be loaded by the threads in one 128-byte transaction (coalesced load). The same holds when accessing NN positions. Each thread will then process, that is perform a MC update on, its corresponding element. Since the updates in all layers follow the same rules, warp-divergence will be a minor problem. Of course, this also means that a part of the dynamics, i. e. the random site-selection, will be the same for all realizations. Since averaging over 32 copies of identical results is pointless, it must be ensured, that the kinetics in all layers is decorrelated by other means.

Random Initial Conditions Using independent initial conditions, all realizations are independent of each other at the start of the simulation, but if identical noise was applied to all of them, they would not form an uncorrelated sample at late times. However, in cases where the evolution of separate systems with different initial conditions is of interest and where the average evolution is of no concern, one may not need to implement other measures. In such cases, having the same, or similar, dynamics act on all realizations may not pose a problem, or may even be strictly required (such as in response calculations). This only hints at applications of the method which were not considered in the present work.

Secondary Noise MC methods using probabilistic updates, like the Metropolis, Glauber or Heat-Bath methods introduce additional noise when evaluating update conditions. In the present implementation, each thread has access to its own independent RNG, thus evaluation of update condition will be independent for each realization. In conjunction with random initial conditions, this should suffice to produce independent samples.

Random Masking The surface growth models considered in the present work always start out from flat initial conditions and do not introduce noise other than by random site selection. Here, the decorrelation can be achieved only updating a random fraction p of the realizations in a multi-surface with each update attempt. This is similar to the way noise is introduced in the SCA approach (section 3.3), only here it only acts on the ensemble, but effectively not on single realizations.

Mixing of Multi-Surfaces Another way to decorrelate realizations arises when simulating more than one multi-surface at once, producing an even larger sample. This is indeed the case in the present implementation in order to maximize device occupation. Since NVIDIA GPUs can only schedule 16 block per multi processor and thus full occupation can only be achieved when running at least 64 threads (two warps) per block, the code will usually only run at maximum efficiency on these devices if at least two multi-surfaces are treated simultaneously.[10]Since there is no reason to coalesce memory accesses across warps, the random site selection is independent for all warps, which thus sample independent multi-surfaces. It would be possible to decorrelate realizations by shuffling them between multi-surfaces during the course of the simulation.

3.4.0.2. Scalar Updates

It has been discussed, how a vector of lattice points belonging to different realizations of a simulation can be updated efficiently, while still obtaining a sufficiently uncorrelated sample. With the SIMD size of 32 and the corresponding SIMD-word-size of 128 bytes, mentioned above, each scalar operation commands 32 bits of data. The goal stated initially was only to have eight bits available per lattice site, which means, that even more complex simulations can be performed using the technique described above than were anticipated, using 32-bit to encode information of one lattice site.

The present applications of this code do not require this much information per lattice site. Currently, the required information can be encoded in about one quarter of each scalar There are two ways in which the remaining memory, and at the same time bandwidth, can be used:

---

[10] This is only a concern in cases where the number of threads that can be run per block is not limited by the number of available registers but only by the maximum number of threads allowed per block.

One way is to encode multiple lattice sites, as described in section 3.1.4, except that here each "spin" uses one byte of memory. In this way each scalar can encode a patch of four ($d = 1$) or 2×2 ($d \geq 2$) lattice sites. This could add memory locality when accessing the NN positions of the selected site, but it would be too small to be treated as a DD cell to perform a whole MC sweep.

A better way is to treat the 32-bit word as another vector lattice sites belonging to four realizations, which increases the number of realizations per multi-surface to $s_{\mathrm{MS}} = 128$. The advantage of this method is that in this way each thread performs four update attempts for every lattice site loaded instead of one. The quadrupled computational load per unit of data will help hide latencies of memory accesses. Since GPUs usually offer a high ratio of compute power over global memory bandwidth, compute-intensive codes can yield better performance than memory-bandwidth-heavy ones. Maximizing the computational load in this way may even yield compute-bound codes for usually not very compute intensive applications.

3.4.0.3. Domain Decomposition

As initially mentioned, application of the MS method serves to eliminate the need for caching DD domains locally for random sampling. Thus MS replaces the second level of DD described in section 3.2.2. Since a single GPU thread does not provide sufficient compute power to perform long simulations on reasonably large systems, DD is still employed to distribute work on replicas among blocks.

The purpose of distributing workload among blocks is twofold:

1. Blocks can be scheduled on different physical multiprocessors, thus using the first layer of physical parallelism on GPUs.
2. In order to maximize device occupancy, more blocks are run than multiprocessors are available. This serves to optimally hide memory latencies by providing several completely independent workloads to each multi-processor.

Based on experience with various types of DD for the octahedron model, presented in sections 4.1.1 and 4.2.1, DT DD with random origin (DTr) is employed here.

Balanced Workload Distribution Contrary to the two-layered approach in section 3.2, the size of DD tiles is not restricted by the size of shared memory per multiprocessor, hence it can be chosen freely. If the choice is not restricted by needs of the calculation, the tile sizes may be optimized to maximize device utilization. The optimal number of tiles, a multiple of the number of multiprocessors, will often not divide the lateral system size. However, a non-uniform partition of the system into this optimal number tiles can always be generated.

Let the desired number of tiles be $\mathcal{T} = \prod_{l=1}^{d} \mathcal{T}^l$, where $\mathcal{T}^l \in \mathbb{N}$ denotes the number of tiles in dimension l and d is the number of dimensions. Ideally $\mathcal{T}^l = \sqrt[d]{\mathcal{T}}$, an approximation to which, satisfying $\mathcal{T}^l \in \mathbb{N}\ \forall l$, can be found using an iterative algorithm. Let further the lateral system sizes be denoted by L^l, then the lateral tile-sizes are given as:

$$t^l_{i_l} = \left\lfloor \frac{L^l}{\mathcal{T}^l} \right\rfloor + \begin{cases} 1 & \text{if } i_l < L^l \mod \mathcal{T}^l \\ 0 & \text{if } i_l \geq L^l \mod \mathcal{T}^l \end{cases} , \tag{3.6}$$

where $i_l \in [0, \mathcal{T}^l)$ is the tile index in dimension l and the brackets $\lfloor\ \rfloor$ denote rounding downwards to the nearest integer. In general, the tiles will then be non-cubic and the lateral sizes of some tiles have to be larger by one lattice site to distribute the remainder of the system, after uniform partitioning.

Limitations While this approach can provide high total performance only a fraction of this ($< 1/128$) is available for each single realization. This limits both system size and achievable simulation time in practice (see next section). Another severe restriction is due to the high memory requirement of at least 128 bytes per lattice site. Even using a K80 or K40 GPU with 12 GB of global memory, the lateral system size is limited to $L < 10000$ in 2 or $L < 460$ in 3 dimensions. Both limitations, however, could be overcome by adding multi-GPU capability. Since DD is already in place, this would just be a matter of distributing tiles over multiple devices and exchanging border information.

3.4.1. Implementation: SkyMC

The above description of the method does also pretty closely describe the present CUDA implementation of the core of the framework called “SkyMC”. Obviously, the above description is also very abstract in that it does not pinpoint the implemented algorithm, the dimension nor what is actually encoded at each lattice point. This does resemble the actual program, since it is pretty much written in this abstract fashion using templates in C++ [112].

When it comes to the method as well as its implementation, these details do not matter. The SkyMC engine provides a way to implement MC simulations on a rectangular lattice in $d \in \mathbb{N}$ using the MS technique as described above on GPU or CPU. The restriction to rectangular lattices stems from the way PBCs are currently implemented. However, any lattice, or graph, which can be mapped onto a rectangular lattice can be treated with little effort.

The properties and performance of the implementations for RSOS and the Potts model shall be discussed briefly in the following. In terms of encoding, both use one byte per lattice site, for an effective multi-surface size of $s_{\mathrm{MS}} = 128$.

3.4.1.1. 2d Restricted Solid on Solid Model

In RSOS, like in the octahedron model, the information required at each lattice site is the height difference to all NN sites. Since this property is antisymmetric for a NN pair, it is sufficient to store only d height differences per lattice site and retrieve the remaining ones from the appropriate neighbors. RSOS has been implemented in 2d only, thus two height differences are stored as two four-bit signed integers (See appendix A.2).

All RSOS runs performed in this work, started from flat initial conditions. The implementation employs *random masking* to produce independent samples within the multi-surface, as introduced for the N=1 RSOS Model in [109]. Two variations of the implementation exist:

1. Each realization has a change p to be updated. This requires one additional random number per update attempt. It also leads to a mild form of warp divergence, because some threads may choose to update all four of their assigned realizations while other may skip all four.

2. Fixing $p = 0.5$ allows employing pairwise masking: Choose one out of each pair of two realizations to update. Since each thread is responsible for four realizations, thus two pairs, it is ensured that all threads will perform exactly two update attempts per iteration, thereby eliminating warp divergence. This is also more efficient in the use of random numbers, since only two random bits are required by each thread per iteration to flip a coin on both pairs.

Variant 2 is considerably faster and thus used in all applications.

3.4.1.2. 2d and 3d Potts Model

The q-state Potts model was implemented for $d \leq 7$ on (hyper-)square lattices, but testing has only been done in two and three dimensions. Each lattice site needs to hold information about the spin species $0 \leq \sigma < q$ occupying the site. Quenched disorder was implemented in the form of a bimodal distribution of random bonds. Thus d bits per site are required to encode the bond-disorder, information about the remaining d bonds is retrieved from appropriate NN sites. Using eight bits per site, this leaves $8 - d$ bits to encode σ, hence restricting $q \leq 2^{8-d}$.

For the decorrelation of realizations, the code relies on the *randomness of initial conditions* and *secondary noise* introduced by evaluating Metropolis update conditions. This turned out to be sufficient for non-conserved dynamics, which is of primary interest in section 5.1. Calculations with quenched disorder profit from the MS approach due to the property that averaging over at least 128 disorder realizations can be done based on a single run.

Kawasaki Dynamics Conserved dynamics has been implemented by way of Kawasaki exchanges [113]. Since in this case, not only the site-selection must be random but also the jump-direction, a direct implementation would again lead to non-coalesced memory accesses to the NN sites of the final position, since it varies across threads. Instead, the implementation generates the same jump direction for Kawasaki exchanges for all replicas in a given warp, but still relies solely on *random initial conditions* and *secondary noise* for the decorrelation of replicas.

Example results are presented in section 5.1.2.

In order to better ensure statistical independence of realizations, this implementation may profit from additional mixing of multi-surfaces in order to avoid correlation of diffusion processes across replicas while still using this efficient approach to implementing Kawasaki dynamics to avoid decoalescence within the MS approach.

3.4.1.3. Sequential CPU Reference

Based on the same memory layout described above, sequential CPU implementations of the presented models were created for reference. A CPU implementation is not required to update multi-surfaces of multiples of 32 scalars, but using multi-surfaces increases the number of cache-hits.

Since each scalar represents another vector of lattice sites belonging to four independent realizations, the implementations each contain an inner loop with fixed range of four over the sub-layers. This loop can be automatically vectorized by the compiler using 128 bit vector registers. This works especially well for the Potts models implementations since the sub-layers are not masked, contrary to the RSOS code.

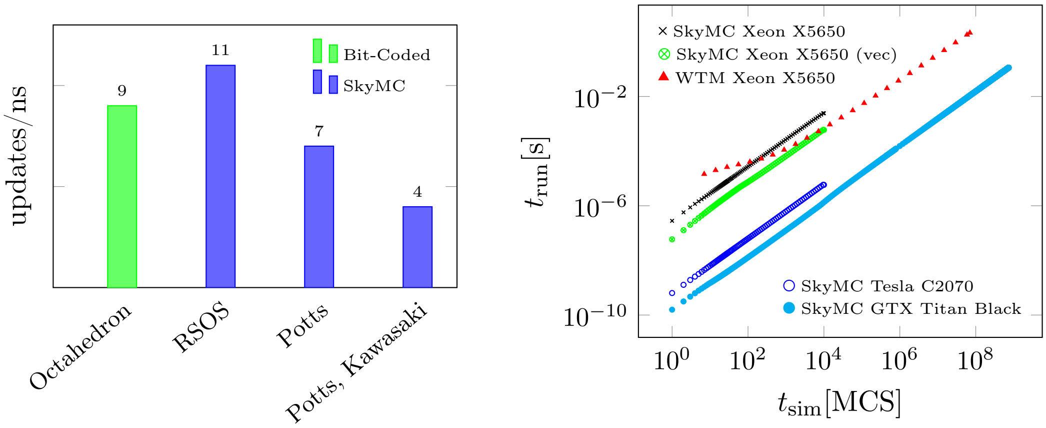
The Potts model implementations can use four way vectorization very efficiently, resulting in a speedup of almost 4× over a non-vectorized version. This is likely to make this implementation more efficient than any conventional (not multi-surface) Metropolis implementation on CPU, despite being not much optimized otherwise.

3.4.2. SkyMC Benchmarks

Figure 3.6a lists the performance of the presented codes based on the SkyMC engine. For RSOS simulation with $N \leq 7$, the achieved performance of RS simulations is even higher than that of the bit-coded implementation of the octahedron model where height-differences are restricted to ± 1. However, in the multi-surface approach on GPU, the performance is spread over $s_{\mathrm{MS}} = 256$ independent realizations, thus if simulations of very large systems over long times are required, a bit-coded simulation may still be preferable for low height-restrictions $N \leq 1$. Profiling on a Tesla K80 shows that this variant is compute-bound (full utilization of arithmetic units) at a global memory throughput usage of $\sim 66\,\mathrm{GB/s}$ (load + store, each contributes half).

For the Potts model implementation, two as well as three–dimensional simulations were considered. Both show about the same performance. The tests with Kawasaki dynamics showed the code running about 10 % faster for 3d compared to 2d systems. This points to remaining optimization potential for the 2d case, since the 3d case is computationally slightly more complex and should therefore be slower to compute.

These benchmarks shall only provide rough estimates for performance. The actual performance can depend on many factors, like system size, simulation time between measurements, total simulation time, even the room temperature during the measurement, since the device may be forced to adjust its clock frequency under



(a) Benchmarks on GTX Titan Black

(b) SkyMC/Metropolis vs WTM

Figure 3.6.: Left(a) Performance of SkyMC codes, when processing $s_{\mathrm{MS}} = 256$ independent realizations, on GTX Titan Black. Performance of the bit-coded octahedron model implementation, for a single realization, (section 3.2.3, 🟩🟩) is given for reference. octahedron and RSOS performance was measured for $2+1$–dimensional systems. For the Potts model two and three–dimensional runs where measured, showing about the same performance. Right(b) Required computation time t_{run} as plotted function of physical simulation time t_{sim} for disordered Potts model simulations with $q = 8$, bond dilution $d = 0.2$ and effective temperature $K = 3K_{c,\mathrm{MF}}$ (see text for details). Comparison is made with a contributed [114] waiting time method (WTM) simulation on CPU. Curves for SkyMC are linear functions, since the performance is independent of the simulation state.

heavy computational load or high ambient temperature. The provided values are time-averages from runs over 10 kMCS.

The presented implementation is not optimized for any specific case: It can handle Potts models for $q \leq 64$ and $q \leq 32$ in 2d and 3d, respectively, with bimodal bond disorder delivering about the same performance. Naturally, special cases could be implemented more efficiently, for example the pure case, without disorder, but this was not done since the Potts model serves primarily as a proof of concept in this work.

The SkyMC engine provides much more flexibility than bit-coded approaches, at at least the same overall performance. The problem with spreading the processing power over many realizations could be compensated by adding multi-GPU capability. With DD already implemented and competitive single-GPU performance, the groundwork for this is already laid.

Rejection-free algorithms At the beginning of of this chapter, rejection-free algorithms have been mentioned. These algorithms become more efficient the fewer Metropolis updates are accepted. Thus they are especially efficient at low temperatures and should become more efficient than any Metropolis simulations at sufficiently late times during coarsening.

A rather recent example from this family of algorithms is the aforementioned waiting time method (WTM) [98], where each degree of freedom, e. g. each spin, in the system is assigned an expected *waiting time* until its next update. Each update is always carried-out on the degree of freedom with the lowest waiting-time which advances the global simulation time by accordingly, before new waiting times for the updated site and the neighbors it interacts with are calculated. Figure 3.6b shows a performance comparison between Metropolis and a sequential implementation a generalized version of the WTM [114]. The simulation time of WTM (▲) is rescaled to match the evolution of a metropolis systems.

The considered system is a non-conservative $q = 8$ state Potts model with bond-dilution, where each bond is broken with a probability $d = 0.2$ in the initial condition (quenched disorder). The effective temperature of the system lies rather deep in the ordered regime with $K = 3K_{c,\mathrm{MF}}$, where $K_{c,\mathrm{MF}} = K_c/(1-d)$ which is the mean-field approximation for the critical effective temperature of the system. The considered system is 2d with a lateral size $L = 128$ ($L = 512$ on GTX Titan Black (●)).

At early times, starting from random initial conditions (quench from infinite temperature), the WTM performs worse than a sequential Metropolis implementation, due to a high demand for book-keeping with updates being likely to be accepted and thus not advancing simulation time a lot. At later times, the book-keeping overhead is overcompensated because it helps to bypass high rejection rates by carrying out rare events which advance simulation time significantly. The performance of the Metropolis algorithm does not change during the evolution of the systems, because each update attempt advances the simulation time by a constant amount, disregarding whether it is accepted.

Since the CPU simulations in this test were run on an older model (Xeon X5650), figure 3.6b includes a plot of the performance on a GPU released at about the same time (Tesla C2070, (○)), where the code delivers about four times less performance than on the more modern GTX Titan Black.

In this specific example is evident that, even though the efficiency of the WTM increases over time, a crossover point with the GPU will not be reached on a relevant timescale. This can be attributed to the quenched disorder in the simulation, which makes the system relax much more slowly and causes many sites to retain high acceptance probabilities for update attempts even at late times. This case serves as an example, that cases do exist, where massively parallel implementations of the Metropolis algorithm can outperform rejection-free algorithms, which cannot be parallelized as efficiently. However, this finding does not mean, that rejection-free algorithms are obsolete: For example in systems without disorder, crossover points are more likely to be found at moderate times.

Figure 3.6b shows two curves for the sequential Metropolis CPU code, where the faster version (⊗) is the result of auto-vectorization by the compiler, which is only possible because of the multi-surface approach.

3.5. Measurements

To derive any information from simulations, observables of the system under investigation have to calculated or *measured*. Observables of interest may be the roughness (2.14) of surfaces or the magnetisation or the degree of phase ordering in phase separating systems as well as two-point functions, like the autocorrelation (4.17) or autoresponse (4.24). In large scale studies, like the surface growth simulations of the octahedron model presented in this work with $V = 2^{32}$ or even 2^{34} lattice sites, it is usually not feasible to store the configuration of the simulated system after each designated measurement interval for later analysis. Instead, the desired observables have to be computed on the fly while the simulation is running.

3.5.0.1. Measurement Intervals

One possible choice for measurement intervals is a constant number of MCS, which is a good choice for sampling within a steady state. Another good choice in a steady state would be random intervals with constant mean. However, this does fit well to the nature of scaling problems, wherein the kinetics slows down with time. Thus a fixed interval length can lead to undersampling at early times and oversampling at late times.

A better choice is to increase measurement intervals over the course of a simulation. All surface growth simulations in this work take measurements at times

$$t_{i+1} = \lceil (t_i + 10) \cdot \mathrm{e}^{m} \rceil \quad \text{, with} \quad t_0 = 0 \quad , \tag{3.7}$$

where the parameter $m > 0$ adjusts the time between measurements; the most common choice in this work is $m = 0.001$. The brackets $\lceil\ \rceil$ denote rounding upwards to the next integer. In order to average many scaling runs, it is imperative to measure at exactly the same times in all simulations, where a recursive rule can cause problems when simulations are interrupted and continued or additional measurements at waiting times for autocorrelation or autoresponse measurements are added. Rule (3.7) was used in related simulations over a long time [27] and was never abolished to keep sequences compatible.

A better rule with a logarithmic scale would be:

$$y = \lfloor \log_{10}(t_i) \cdot \widetilde{m} \rceil \tag{3.8a}$$

$$t_{i+1} = \begin{cases} 10^{y/\widetilde{m}} & \text{if } 10^{y/\widetilde{m}} > t_i \\ 10^{(y+1)/\widetilde{m}} & \text{if } 10^{y/\widetilde{m}} \leq t_i \leq 10^{(y+1)/\widetilde{m}} \\ t_i + 1 & \text{otherwise} \end{cases} \tag{3.8b}$$

This rule reproduces the same sequence irrespective of the initial value t_0. The parameter $\widetilde{m}$ is the number of samples to be taken per decade. The first condition in equation (3.8b) would be obsolete mathematically if one rounded down ($\lfloor\ \rfloor$) in equation (3.8a), instead of rounding to the nearest integer ($\lfloor\ \rceil$). However, in that case the expressions would be less stable against floating-point errors in numerical calculations.

3.5.0.2. Measuring using Heterogeneous Resources

In the GPU implementations presented in this chapter, all the computational load for the actual simulation is on the GPU, letting all CPU resources stay mostly idle. Thus, the algorithms for taking measurements are executed in parallel with the simulation running on GPU. The implementations of the measurement algorithms are multithreaded, to utilise multiple CPU cores which are usually available per GPU. In this way the codes developed and used in this work make use of the full heterogeneous parallel environment provided by CPUs and GPUs installed in the same machine.

Offloading measurements to the CPU is only efficient as long the CPU is able to complete this work before the GPU completes the next interval of the simulation. When measurements are very frequent, such as at early times during a scaling run, this condition may not be met, which makes pausing the simulation and running the analysis on the GPU preferable. This problem is very prevalent in simulations using the non-local SCA implementation presented in section 3.3.2.

A good solution is a dynamic load balancing approach, where a measurement is offloaded to the CPU if it is currently idle, but performed on the GPU, if the CPU is still busy performing the previous measurement. This has only been implemented for Potts model simulations in SkyMC.

4. Monte-Carlo Investigation of the Kardar–Parisi–Zhang Universality Class

Parts of sections 4.1.1 and 4.2 have been adapted into a submitted manuscript [115] and a preprint [116].

The various massively parallel simulation approaches presented in the previous chapter enable large-scale simulations leading to new insights about surface growth. The primary goals of this chapter are twofold: First, to test the impact of these implementations on dynamical growth properties and secondly, to provide more precise numerical estimates of universal exponents and, to a small extent, of scaling forms. The first goal is divided into two parts, relating to RS and SCA dynamics, respectively. The former is to be benchmarked against really sequential simulations as reference, which they should ideally reproduce closely. The SCA dynamics is then benchmarked against RS simulations to discern the impact of the controllable correlation it introduces.

Simulations and results for scaling properties of the $(2+1)d$ KPZ universality class are presented and discussed in section 4.1. The section closes with a summary because some results are required as a basis for the following sections. Dynamical properties for instance aging related to autocorrelation and autoresponse in the growth regime are investigated in sections 4.2 and 4.3, respectively.

In (1+1) dimensions, many properties of the KPZ equation (2.12) can be calculated analytically. Even though surface growth is a non-equilibrium process there is a fluctuation-dissipation relation (FDR) present in $(1+1)d$ [61,117]:

$$T\chi(t,s;r) = -\partial_r^2 C(t,s;r) \tag{4.1}$$

where χ and C are the autoresponse and autocorrelation functions, respectively. r is the spatial coordinate, which will be integrated over later to arrive at equations for the two functions, equation (4.17) and (4.24), respectively. t and s denote the simulation time and waiting time, respectively. This relation is a result of time-reversal symmetry being present in the $(1+1)d$ KPZ universality class.

This FDR fixes the dynamical and the roughness exponents in $(1+1)d$ to $z_{1d} = 3/2$ and $\alpha_{1d} = 1/2$, respectively. Due to the definition of the dynamical exponent, the growth exponent is then given as $\beta_{1d} = \alpha/z = 1/3$. Similar results were not obtained analytically in the $(2+1)d$-case, primarily because the strong-coupling KPZ regime is not accessible for perturbational methods.

Aging in growing KPZ systems in $(1+1)d$ has been studied recently [118]. From the FDR, the relations $\lambda_C = \lambda_R$ and

$$1 + a = b + 2/z \tag{4.2}$$

were found and confirmed by simulation results, where a and b are the aging exponents for autocorrelation and response, respectively. No such relation is expected to hold in higher dimensions, such as in the $(2+1)d$ case discussed in sections 4.2 and 4.3. However, this topic will be revisited in section 4.4, where this chapter is summarized.

4.1. Evolution of Surface Roughness

In two dimensions, as well as in higher dimensions, not much is known analytically about the scaling behavior of the KPZ equation (2.12). One relation which must be fulfilled in all dimensions is the scaling relation based on the Galilean symmetry [61]:

$$2 = \alpha + z = \alpha\,(1 + 1/\beta) \tag{4.3}$$

This equation relates the roughness exponent to the dynamical or the growth exponent and thus allows one to check numerically obtained estimates for consistency.

Based on restricted solid-on-solid model (RSOS) simulations in various dimensions, Kim and Kosterlitz (KK) conjectured general forms for the universal exponents for all $d \in \mathbb{N}$: [71]

$$\begin{aligned} \beta(d) &= 1/(d+2) \\ \alpha(d) &= 2/(d+3) \end{aligned} \tag{4.4}$$

and consequently

$$z(d) = 2(d+2)/(d+3)$$

The KK conjecture has been tested in multiple studies studies of surface growth models [27, 79, 119, 120] which found smaller growth and roughness exponents than conjectured. In a very recent study based on RSOS simulations for various height restrictions N in $(2+1)d$, Kim concluded that the conjecture had only been violated in earlier studies solely because of too strong restrictions of the height differences between NN sites [83]. An example are large-scale studies of the octahedron model [27], where $\Delta h = \pm 1$. This proposition is tested in section 4.1.2.

Apart from the exponents, the shapes of the rescaled width and height distributions of the interface $\Psi_L(\varphi_L)$ were shown to be universal in KPZ models [121, 122]. Here, L refers to the system size, to which the distributions are sensitive in the steady state. φ_L denotes the interface observable in question: Width W^2 or height h. The non-rescaled probability distributions are denoted by $P_L(\varphi_L)$. Without rescaling, the moments are defined as:

$$\Phi_L^n[\varphi_L] = \int_0^\infty (\varphi_L - \langle\varphi_L\rangle)^n \, P_L(\varphi_L) \, \mathrm{d}\varphi_L \quad , \tag{4.5}$$

Two standard measures of the shape, the skewness

$$S_L[\varphi_L] = \langle\Phi_L^3[\varphi_L]\rangle / \langle\Phi_L^2[\varphi_L]\rangle^{3/2} \tag{4.6}$$

and the kurtosis

$$Q_L[\varphi_L] = \langle\Phi_L^4[\varphi_L]\rangle / \langle\Phi_L^2[\varphi_L]\rangle^2 - 3 \quad , \tag{4.7}$$

are usually calculated, often in the steady state. Both measures are invariant under the rescaling required to produce the universal form and are thus universal themselves. The universal, rescaled forms are:

$$\Psi_L[W^2(L)] = \langle W^2(L)\rangle P_L(W^2(L)/\langle W^2(L)\rangle) \tag{4.8}$$

for the width and

$$\Psi_L[h_L(r)] = L^\alpha P_L(h_L(r)/L^\alpha) \tag{4.9}$$

for the surface height. Note, that $\langle h_L\rangle \equiv \Phi_L^0[h_L] \equiv 0$ in the co-moving frame of the surface.

In the following, first scaling in the $(2+1)d$ octahedron model and the dependence upon different site-selection dynamics is discussed (section 4.1.1). Simulations of the $(2+1)d$ RSOS model including $N > 1$ are presented in section 4.1.2 to address the issue raised by Kim [83]. The final results for universal KPZ scaling exponents in $(2+1)$ dimensions are summarized and discussed in section 4.1.3.

4.1.1. Comparison of Parallel Implementations of the Octahedron Model

When an algorithm is implemented in an approximate way, such as using DD to retain RS dynamics, or is basically replaced by another, such as SCA, the result of the new implementation needs to be checked for statistical accuracy explicitly. The ideal way to do this would be to compare with exact results, but no analytical solutions for 2d KPZ are available. A method used before, in [27,103], is to compare with results produced by the original, that is the sequential CPU, implementation. This, however, does only allow for strong conclusions up to the accuracy of the result that can be produced with the original, sequential implementation, while the point of using a more efficient algorithm is to obtain more precise results. Many physical systems can be well approximated linearly in the vicinity of a known point. Thus, if the results of the new algorithm agree with those of the original one up to the latter one's accuracy, it is in general a good first assumption, that no major deviations occur, when going beyond that by just one further digit.

Since asynchronous updates using DD necessarily introduce approximations with respect to the real RS algorithm, the following questions arise:

- Of what order of magnitude are the intrinsic errors? Can this be controlled?
- Do deviations only affect non-universal properties?

The primary free parameter of DD is the size of decomposition domains. The aspect ratio of domains turns out to be of secondary importance. Answers to these questions can be found by performing a convergence study for the primary parameter and checking for self-consistency of results produced by the new algorithm with different DD parameters.

4.1.1.1. The Growth Regime

Random Sequential Dynamics Considering the roughness scaling of the KPZ surface, one immediately finds a non-universal quantity which is sensitive to DD. The left plot in figure 4.1 shows only the non-universal part of the surface roughening under different DD schemes, by dividing the scaling function by the universal PL. All curves will eventually approach a constant value when they cross over to the asymptotic power-law. This non-universal roughness coefficient clearly depends on the configuration of DD domains and is related to the amplitude of the noise (2.13) which is left as a free parameter in the in the KPZ equation (2.12). In the simulation, the noise is produced by random site selection. When domain decomposition is introduced the variance of the rate at which each site is selected for updates decreases with the size of the smallest decomposition domains. In other words, the system is sampled more smoothly with fine DD, which slightly inhibits the roughening of the surface, even when updates are uncorrelated. The plot shows, that the roughness produced by the GPU implementation evolves with a different constant factor than that of the sequential code, which does not use DD. The apparent ordering of the roughness coefficient by the volume of block-layer domains is consistent with the hypothesis of an implementation-dependent noise amplitude D.

The plot contains two lines (-·-·) with the same configuration of block-layer domains (`TC=2,2`), but the volume of device-layer domains differs by a factor of four: `T=5,4` vs. `T=4,3`. Still, both curves are almost identical, which suggests that the influence of DD at device-layer on site-selection noise is negligible in the presence of a second layer with smaller domains. In this section, at device-layer only DTr will be considered for its correlation properties, which are discussed in section 4.2.1.

The most striking difference between single-hit DT-based schemes at block-layer (DTrDT, DTr at device level and single-hit DTr at block level (DTrDTr)) and the sequential code (CPU) is the local minimum which DT shows for small TC. This

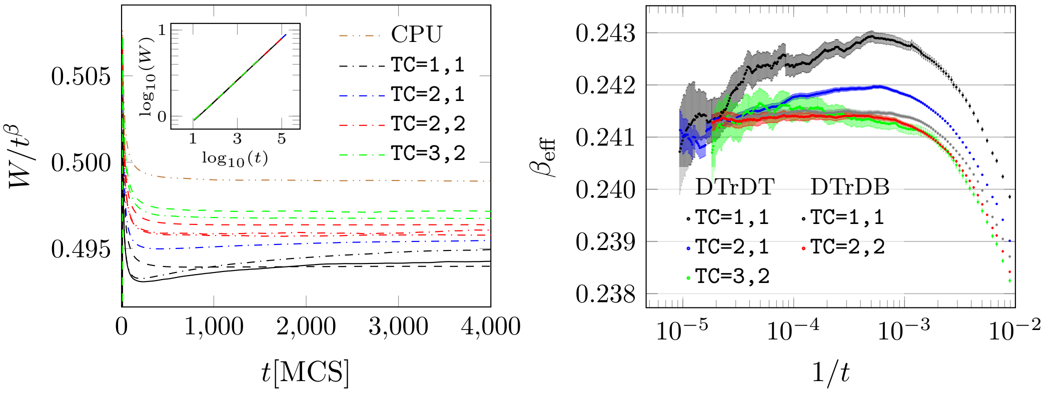


Figure 4.1.: Left: Scaling functions under RS dynamics divided by the universal power scaling PL as a measure of the amplitude of site selection noise. Colors correspond to TC configurations. Line styles indicate DD scheme: (-·-) DTr at device level and single-hit DT at block level (DTrDT), (—) DTr at device level and single-hit DTr at block level (DTrDTr) and (- - -) DTr at device level and single-hit DB at block level (DTrDB). Two lines (-·-) (DTrDT) show configurations `TC=2,2 T=5,4` and `TC=2,2 T=4,3` (see text for details). Inset: Plot of the roughness scaling. Right: Effective exponents for selected DD configurations. System sizes are $L_\bullet = 2^{17}$ others $L = 2^{16}$. Sample sizes are: $n_\bullet \geq 85$, $n_\bullet \geq 396$, $n_\bullet \geq 89$, $n_\bullet \geq 1107$ and $n_\bullet \geq 708$. Propagated 1σ error bars are attached to the effective exponents, merging into an error-corridor at late times due to the dense placing of points.

is also not present using single-hit DB at block-layer (DTrDB) which seems to converge monotonously toward the asymptotic value from above, like the sequential implementation does.

Considering that the lateral sizes of block-layer domains using DT correspond only to half TC, while they are identical for DB, uncovers another detail about DT: The block-layer domains of DTrDT, `TC=2,2` (-·- in figure 4.1, left) have the same size as those of DTrDB, `TC=1,1` (- - -). Thus, since the block-layer domains are the smallest DD units in both cases, the roughness coefficients of these curves should be comparable. Instead, the roughness coefficient of DTrDT, `TC=2,2` is closer to that of DTrDB, `TC=2,2` (- - -), which has the same TC, but a fourfold larger block-layer domain. This suggests, that the site selection noise under single-hit DT is governed by the size of the TC, rather than that of the block-layer domains, although not perfectly. This can be made intuitively plausible by considering, that, for each update attempt, a thread effectively selects a site at TC-scope, with no a priori restriction to a block-layer domain. This way, single-hit updates neglect borders between block-layer domains within a TC.

This does not hold for single-hit DT with very small TCs where the roughness coefficient shows a minimum at early times and tends towards a larger value than expected based on this hypothesis (compare DTrDT, `TC=1,1` (-·-·-) or DTrDTr, `TC=1,1` (——) and DTrDB, `TC=1,1` (- - -)). This finding suggests, that single-hit DT introduces some additional, correlated noise which is strong enough to be seen for small TC, while single-hit DB does not.

The right panel in figure 4.1 shows effective scaling exponents for a selected set of DD configurations. All curves suggest the same asymptotic value for β to a better accuracy than the most precise GPU results published in [27]. The simulations differ only in corrections to scaling. Most notably, the effective exponents from DTrDB runs show a plateau over almost two decades, suggesting that no significant corrections are present at late times. Averaging these plateaus (i. e. extrapolating to infinity with a constant), yields estimates for the scaling exponent of $\beta_{\mathrm{DTrDB},\texttt{TC=1,1}} = 0.241\,46(1)$ and $\beta_{\mathrm{DTrDB},\texttt{TC=2,2}} = 0.241\,39(1)$. The assumption of two different scaling exponents would place the calculations with different DD cell sizes in different universality classes. This is much less likely than small corrections being present, which have just been ignored by assuming the plateau to be constant. Taking unknown corrections into account a unified estimate of $\beta = 0.2414(2)$ seems appropriate. This error margin is of about the same size as the 1σ-error bars attached to the effective exponents at late times.

The effective exponents of DTrDT simulations using large `TC=3,2` do also suggest reduced corrections and agree with this estimate. Since DTrDB offers low corrections already at smaller TC it is the superior method regarding scaling properties.

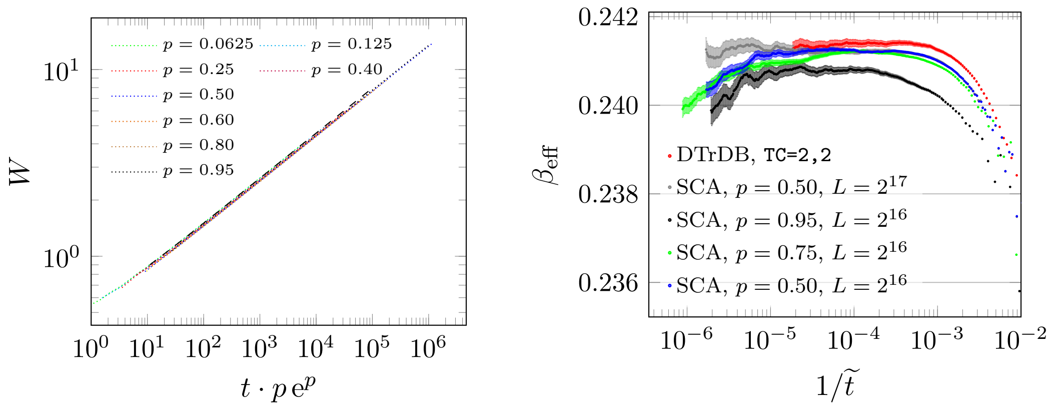


Figure 4.2.: Left: Width-scaling plot under SCA dynamics. Curves are collapsed over p by rescaling time as $\widetilde{t} = t \cdot p\,\mathrm{e}^p$. Scaling under RS dynamics is shown for comparison (- - -, DTrDB, `TC=2,2`). Right: Effective scaling exponents under SCA dynamics for $p = 0.95$ ($n_\bullet \geq 2254$), $p = 0.75$ ($n_\bullet \geq 6430$) and $p = 0.5$ ($n_\bullet \geq 373$, $n_\bullet \geq 3062$). RS data is shown for comparison (●). Propagated 1σ error bars are attached to the effective exponents, merging into an error-corridor at late times due to the dense placing of points.

Stochastic cellular automaton Figure 4.2 displays scaling data obtained from SCA simulations. Here, the update probability $p > 0$ governs simulation results, assuming $q = 0$. If $p < 1$, on average only a fraction p of lattice sites will be updated during one sweep. This suggests that the simulation time is being rescaled linearly with p, which is true for RS dynamics. However, in case of SCA such a linear relation cannot hold, since at $p = 1$ no roughening would occur at all. Instead, with flat initial conditions, the system would oscillate between two flat states. To find the dependence of the time scale under SCA on p, the linear ansatz needs to be extended with a non-linear factor. Empirically, for $p \leq 0.95$, the non-linear part seems to be well approximated by an exponential. The form

$$\widetilde{t}(p) = t \cdot p\, \mathrm{e}^{p} \tag{4.10}$$

yields a reasonable collapse of the scaling functions, as shown in the left panel of figure 4.2, which displays data from DTrDB RS runs for comparison.

The right panel of figure 4.2 shows effective scaling exponents from SCA simulations at $p = 0.5$ and 0.95, where time has been rescaled according to equation (4.10). Like in the RS case, there is a plateau spanning almost two decades in the $L = 2^{16}$ datasets, suggesting significantly different scaling exponents depending on p: $\beta_{p=0.95} = 0.240\,79(1)$ and $\beta_{p=0.5} = 0.241\,22(1)$. While the latter falls within the error margin of the value derived from RS runs, the former does not. The spread of these values $\Delta\beta_{\mathrm{SCA}} = 0.000\,43(2)$, which is marginally significant even though corrections have not been considered. The effective exponents for SCA runs with $p = 0.75$ show some intermediate behavior: While they are close to $\beta_{\mathrm{eff},p=0.5}(1/\widetilde{t})$ for $1/\widetilde{t} \gtrsim 1 \times 10^{-4}$, they tend downward after that, approaching $\beta_{\mathrm{eff},p=0.95}(1/\widetilde{t})$.

The possibility, that SCA simulations may exhibit different growth exponents depending on $p < 1$ may raise doubts about them correctly modelling KPZ surface growth. If this is the case, then the asymptotic scaling exponents under SCA dynamics may differ from the actual KPZ value and converge only in the RS-limit $p \to 1/V$.

The plots also show the effective exponents decreasing at late times. Such behavior may be attributed to the onset of the steady state, which seems unlikely to be the case here: The steady state should not be reached until about one decade after the end of the displayed plot. This can be estimated from a finite size scaling collapse as can be found in [27,74] or from figure 4.5 (left). Another explanation might be a possible building-up of blockades in the dimer lattice-gas due to the correlated updates, which start to move through the system as waves, and slow down the growth of surface roughness. This effect would be assumed to depend on p and to be independent of system size, although for smaller systems real finite size effects may set in before this phenomenon could be observed. The presented data shows, that it does scale with p, since the visible kink appears at about at the same rescaled time $\widetilde{t} \approx 1.7 \times 10^{5}$ MCS in all SCA runs at size $L = 2^{16}$, except for $p = 0.75$. This feature may be hidden by the much longer downwards slope in $\beta_{\mathrm{eff},p=0.75}(1/\widetilde{t})$, providing no reason to assume that the case $p = 0.75$ does not share the underlying phenomenon with the others.

Another, although less likely, hypothesis is that this is actually a sign of more

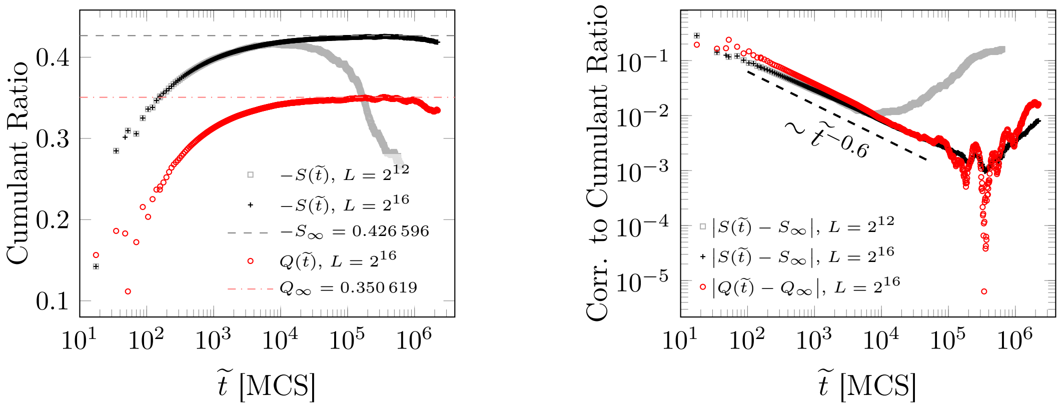


Figure 4.3.: Skewness S (+) and kurtosis Q (○) of the distribution of interface heights in the growth regime. The data belongs to the set of SCA runs with $p = 0.75$, $L = 2^{16}$ ($n_{●} \geq 6430$, compare figure 4.2). The skewness for a smaller dataset for $L = 2^{12}$ ($n_{□} \geq 45$) is included to illustrate finite-size behavior. Left: Cumulant ratios as functions of time. The horizontal lines show the obtained fit parameters for the asymptotic values, to guide the eye. See text for proper values with error estimates. Right: Finite-time and finite-size corrections to the asymptotic values of the cumulant ratios.

complex corrections, which may include oscillations at late times, allowing for the correct universal exponent to be reached asymptotically for all p.

An, as of yet inconclusive, attempt has been made to check if the observed veering-down is a finite-size effect: The presented dataset (●) for $L = 2^{17}, p = 0.5$ does not seem to veer down like its $L = 2^{16}$-equivalent, but is oscillating. The oscillation may be noise within the statistical error. The presence of a kink can be excluded up to a 1σ-error, which is a weak indication of a finite-size effect. Conclusive data would require highly accurate, and thus expensive calculations at different system sizes. These have not been performed because the presented high-quality datasets are primarily the basis of the SCA aging studies presented in section 4.2.3, where studying one system size with high precision suffices.

4.1.1.2. Distribution of Interface Heights in the Growth Regime

For the SCA dataset with $p = 0.75$ (●), the first seven moments of the height distribution have been calculated in order to obtain information about the shape of the distribution of interface heights. Figure 4.3 shows the time-evolution of the cumulant ratios S and Q, defined by equations (4.6) and (4.7). Both values approach their respective asymptotic values for the growth regime following a PL, but move away again at late times. The values S_∞ and Q_∞ can be determined by performing a fit of the form

$$R(t) = R_\infty + b_R \cdot \widetilde{t}^{c_R} \quad ,$$

where R is a placeholder for S or Q. Considering the interval $200 \leq \widetilde{t} \leq 200\,000\,\mathrm{MCS}$ for the fit, excludes both the strong oscillations at early times and the departure at later times. This yields $S_{\text{heights,growth}} = -0.427(2)$ and $Q_{\text{heights,growth}} = 0.351(3)$. Both absolute values are in good agreement with literature values [123, 124] for the KPZ universality class. The sign of S depends on the choice $p \gtrless q$ in the simulations, which determines the sign of the parameter λ in the KPZ equation (2.12).

The right panel of figure 4.3 shows the corrections to these asymptotic values at early and late times. The error estimates given above originate from this representation: The error is assumed to be on the order of the closest approach of the numerical data to the asymptotic value. The exponents for the approach of the cumulant ratios at finite times the asymptotic values are $c_S \approx -0.54$ and $c_Q \approx -0.60$. The value $c_R \approx -0.6$ also holds for a number of other dimensionless cumulant rations, which were calculated, but are not displayed here.

After the closest approach to the asymptotic values in the growth regime, $S(\widetilde{t})$ and $Q(\widetilde{t})$, both, move in the direction of their respective values in the steady state: $S_{\text{heights,steady}} \approx 0.26$ and $Q_{\text{heights,steady}} \approx 0.13$ [109, 125, 126]. The shape of the distribution of surface heights changing in this way is an indication of finite-size effects becoming relevant at $\widetilde{t}_{\text{fs}} \approx 3 \times 10^5\,\mathrm{MCS}$. This coincides with the time at which the kink[1] in $\beta_{\text{eff}}(\widetilde{t})$ is observed for SCA runs at $L = 2^{16}$ in figure 4.2, right. Hence it becomes clear, that this change in β_{eff} is caused by finite-size effects. In the figure, the finite size behavior of the skewness is illustrated by a similar plot for a smaller system size $L = 2^{12}$, where the steady state is reached at $\widetilde{t} < 4 \times 10^5\,\mathrm{MCS}$.

4.1.1.3. KPZ Ansatz for the Growth Regime

Analytical and numerical investigations of KPZ models in 1+1 dimensions found that finite-time corrections to $h(t)$ took the form $\sim t^{-\beta}$ for the interface height [127–129]:

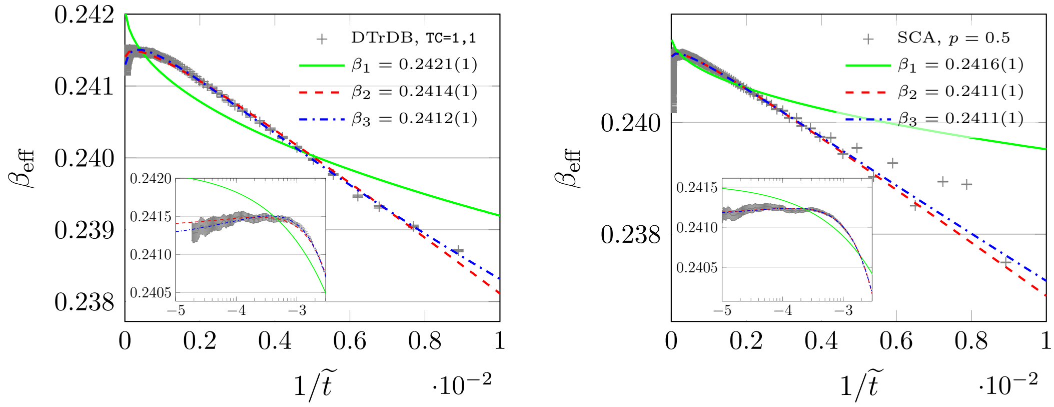
$$h(t) = \operatorname{sign}(\lambda) \cdot (\Gamma t)^{\beta} \chi + \xi + \zeta t^{-\beta} \quad ,$$

where λ, Γ, ξ and ζ are model-dependent parameters and χ is a universal random variable distributed following the Gaussian orthogonal ensemble (GOE), in the present case of a flat geometry. The KPZ ansatz hypothesis states, that a generalisation of this form should also hold in higher dimensions [124], leading to the following ansatz for the effective growth exponent:

$$\beta_{\text{eff}} = \beta + \sum_{n=1}^{N} c_n t^{-n\beta} \quad , \tag{4.11}$$

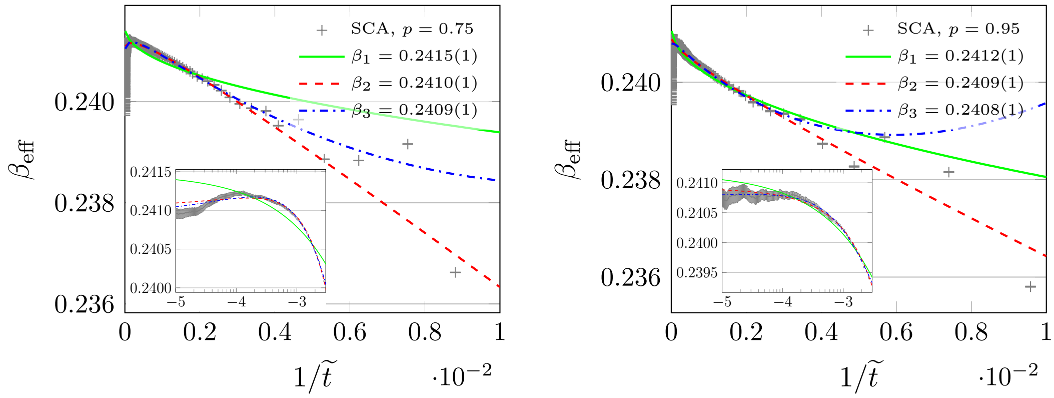
---

[1] The kink is observed at $\widetilde{t}' \approx 1.7 \times 10^5\,\mathrm{MCS}$, roughness values at two different times contribute in the calculation of $\beta_{\text{eff}}(\widetilde{t}')$: $\widetilde{t}_1 \approx 2 \times 10^4\,\mathrm{MCS}$ and $\widetilde{t}_2 \approx 3.2 \times 10^5\,\mathrm{MCS} \gtrsim t_{\text{fs}}$.



(a) RS: DTrDB, `TC=1,1`

(b) SCA: $p = 0.5$



(c) SCA: $p = 0.75$

(d) SCA: $p = 0.95$

Figure 4.4.: Effective exponents β_{eff} for roughness growth with KPZ ansatz fits using the form (4.12) to orders one through three. The resulting asymptotic values for β are given in the legends accompanied by the uncertainty of the fit parameter. The insets show a zoom to the late-time region $1 \times 10^{-5} \leq 1/\widetilde{t} \leq 3 \times 10^{-3}$, cutting-off before the kink visible in figure 4.2, right. The $1/\widetilde{t}$ axes are scaled logarithmically using base 10 and are labeled only with the exponents to improve readability. Panel (a) shows the RS dataset using DTrDB with `TC=1,1`. Fits were performed in the interval $1 \times 10^{-5} \leq 1/\widetilde{t} \leq 1 \times 10^{-2}$. Panel (b)-(d) show SCA datasets with $p = 0.5, 0.75$ and 0.95, respectively. The fits were restricted to the interval shown in the inset. See the captions of figures 4.1 and 4.2 for sample sizes.

with non-universal parameters c_n and N. Higher moments of the height $\langle h^n \rangle$ show corrections $\sim t^{-n\beta}$, accordingly, and thus $\sim t^{-2\beta}$ for the roughness, prescribing:

$$\beta_{\mathrm{eff},W} = \beta + \sum_{n=1}^{N} c_n t^{-2n\beta} \tag{4.12}$$

In the $2+1$–dimensional RSOS model, the dominant corrections to the roughness growth were found to be of order $\sim t^{-4\beta}$ [124], which motivates including more than just the leading orders in these forms. Ideally, such a model would fit the data well as soon as all relevant orders are included. Adding more terms should not improve the fit quality then. However, with noisy data, adding more free parameters in this way, can result in over fitting, if not in convergence-problems.

Figure 4.4 shows fits of equation (4.12) to previously introduced datasets, see figure and corresponding caption for details. It is immediately apparent, that model (4.12) with $N = 1$ does not describe the presented data, the $n = 2$-term is required, same as for the RSOS model.

In case of RS simulations, the model appears to fit reasonably well rather early times $\widetilde{t} \geq 100$. SCA runs on the other hand show strong oscillations at early times, which are beyond the type of model suggested by the KPZ ansatz. Still, the model fits a late time regime in the interval $1 \times 10^{-5} \leq 1/\widetilde{t} \leq 3 \times 10^{-3}$. The case $p = 0.75$ poses an exception here, due to its non-monotonous characteristics in the apparent cross-over from $p = 0.5$-like to $p = 0.95$-like behavior.

A more quantitative view is provided by table 4.1, which lists the reduced sums of residuals χ_{red} to judge the agreement between model and data. Equation (4.12) describes all datasets best, if two terms $\sim \widetilde{t}^{-2\beta}$ and $\sim \widetilde{t}^{-4\beta}$ ($N = 2$) are included, except SCA, $p = 0.75$, which requires more free parameters to approximate its more complicated form at late times.

Where the KPZ ansatz does indeed apply, fits of the more general models (4.11) should not show increased agreement with the data. The table shows them to be less consistent with respect to the resulting estimates for β. They provide the best description of the data with only the term $\sim \widetilde{t}^{-2\beta}$ but one or two additional *odd* terms present ($N = 2, 3$). The best fits resulting from models (4.12) are consistently better than those of (4.11), across all datasets, which justifies discarding the latter class of models and thereby supports the KPZ ansatz hypothesis.

To obtain estimates for β for each dataset, the best-fit value is used. The spread of values for β for higher values of N, provides an estimate of the potential for over fitting present in the model, which may also be a reasonable error estimate for a small confidence interval of 1σ. For simulations with RS dynamics, this yields $\beta = 0.2414(2)$, which is remarkably identical to the previous estimate found in section 4.1.1.1, based on an average of the late-time plateau exhibited by β_{eff}.

The estimates obtained this way may be less reliable for SCA datasets, because here the model can only fit a fraction of the available time series which must exclude the initial oscillations. It can be noted, that the estimate $\beta_{p=0.5} = 0.2411(2)$ is

Table 4.1.: Fit parameter β for the extrapolation of β_{eff} using the KPZ ansatz in the most general form (4.11) and in the form more suitable to describe corrections to roughness-scaling, equation (4.12), both with maximum orders one through six. Error margins given are uncertainties of the fit parameters, not actual error estimates for β. The reduced sums of residuals χ_{red} are given to judge the quality of the fits. For each dataset, the value of χ_{red} closest to one, which indicates the best agreement achieved between data and model, is underlined.

	RS				SCA					
	DTrDB, `TC=1,1`		DTrDB, `TC=2,2`		$p = 0.5$		$p = 0.75$		$p = 0.95$	
N	β	χ_{red}	β	χ_{red}	β	χ_{red}	β	χ_{red}	β	χ_{red}
					equation (4.11)					
1	0.2430(1)	10.04	0.2433(1)	9.24	0.2420(1)	7.03	0.2419(1)	8.63	0.2418(1)	3.15
2	0.2396(1)	1.82	0.2396(1)	1.29	0.2403(1)	1.62	0.2400(1)	1.31	0.2403(1)	0.51
3	0.2411(1)	0.79	0.2408(1)	0.55	0.2414(1)	0.64	0.2404(1)	1.18	0.2401(1)	0.49
4	0.2421(2)	0.67	0.2421(1)	0.32	0.2414(1)	0.64	0.2385(1)	0.63	0.2403(2)	0.48
5	0.2386(3)	0.47	0.2409(2)	0.28	0.2388(3)	0.51	0.2394(3)	0.61	0.2399(5)	0.48
6	0.2377(8)	0.47	0.2403(5)	0.28	0.2333(9)	0.45	0.2420(7)	0.59	0.2485(8)	0.40
					equation (4.12)					
1	0.2421(1)	7.28	0.2422(1)	6.27	0.2416(1)	5.22	0.2415(1)	6.54	0.2412(1)	1.76
2	0.2414(1)	1.16	0.2414(1)	1.10	0.2411(1)	0.65	0.2410(1)	1.72	0.2409(1)	0.81
3	0.2412(1)	0.65	0.2412(1)	0.33	0.2411(1)	0.64	0.2409(1)	1.51	0.2408(1)	0.52
4	0.2413(1)	0.53	0.2412(1)	0.29	0.2412(1)	0.59	0.2407(1)	0.86	0.2407(1)	0.50
5	0.2412(1)	0.51	0.2412(1)	0.28	0.2412(1)	0.59	0.2406(1)	0.68	0.2405(1)	0.41
6	0.2412(1)	0.51	0.2412(1)	0.28	0.2411(1)	0.57	0.2403(1)	0.50	0.2407(1)	0.38

basically the same as the one previously obtained. For large p the estimate $\beta_{p=0.95} = 0.2409(4)$ now agrees with the previous one as well as, marginally, with the one for RS runs. However, this result is obtained because the model predicts the effective exponents for move upwards for $\widetilde{t} \to \infty$. This prediction should be regarded with care, since it remains unclear how far the KPZ ansatz remains valid for large $p \to 1$. For $p = 0.75$, it appears prudent to refrain from making an estimate because the model does not actually describe the data well.

4.1.1.4. The Steady State

Finite size scaling in the octahedron model is as such not a subject of the present study, thus dependencies on DD were not studied in detail. It is conceivable, that the above arguments regarding the noise amplitude do apply here, too, but, based on the observations presented above, the effect can be expected to be rather small. This still implies, that when performing a finite-size study, the size of domains needs to be kept constant across all simulations (all L). Large samples for small system sizes are required for this effect to be quantified, which was not attempted for the octahedron model.

The best estimate of the roughness exponent from octahedron model calculations therefore remains $\alpha = 0.393(3)$ [27]. Using the above estimate $\beta = 0.2414(2)$, yields

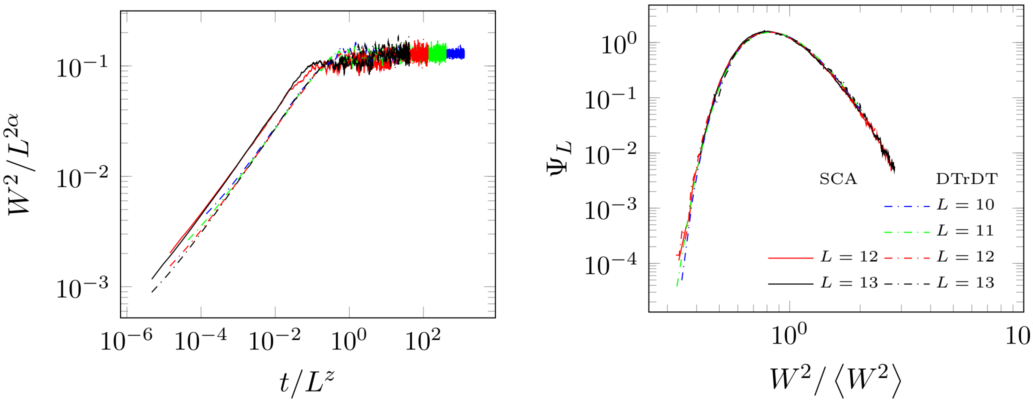


Figure 4.5.: Left: Scaling collapse into the steady state of data from SCA ($p = 0.95, q = 0$, solid lines) and DTrDT (dash-dotted lines) simulations. SCA introduces non-universal corrections to scaling. Points in the steady state ($t > 2\,\mathrm{MMCS}$) are averaged over $\Delta t = 50\,\mathrm{kMCS}$. Key in right panel applies. Right: The universal scaling function Ψ_L is identical for SCA and RS dynamics.

a dynamical exponent $z = 1.63(2)$. A scaling data collapse into the steady state based on these parameters is shown in the left panel of figure 4.5. Dashed curves represent RS simulations using DTrDT, `TC=2,1`, solid lines SCA with $p = 0.95, q = 0$. Both sets collapse well separately. The difference in the growth regime is due to the modified timescale in SCA simulations (the time axis in the plot is *not* rescaled to compensate). Apparently, SCA dynamics leads to the reduced stationary width.

The presented data is only averaged over few (8 - 13) samples, but the simulations are long and do thus reach far into the steady state. Averaging over time in the steady state allows computing the width distribution $P_L(W^2(L))$ rather precisely. The universal scaling function of the width $\Psi_L(W^2(L))$, equation (4.8), is plotted for both RS and SCA dynamics in the right panel of figure 4.5. Even though $\langle W^2_{\mathrm{SCA}}\rangle_L < \langle W^2_{\mathrm{RS}}\rangle_L$, the distributions turn out identical independently of the employed type of dynamics. This is evidence that the octahedron model under SCA does fall in the KPZ universality class.

4.1.2. Investigations using RSOS

A modified version of this section has been published as an article [130].

4.1.2.1. The Growth Regime

Roughening of $2+1$–dimensional RSOS surfaces was studied for restriction parameters $N = 1, 3, 5, 7$, starting from flat initial conditions. To obtain estimates for the exponent β, the growth of surfaces was followed up to $t = 1 \times 10^5$ MCS, which is well before saturation becomes an issue at the investigated system sizes $L = 4096, 8192$ and 9605. The largest system size was bounded by memory constraints, filling up

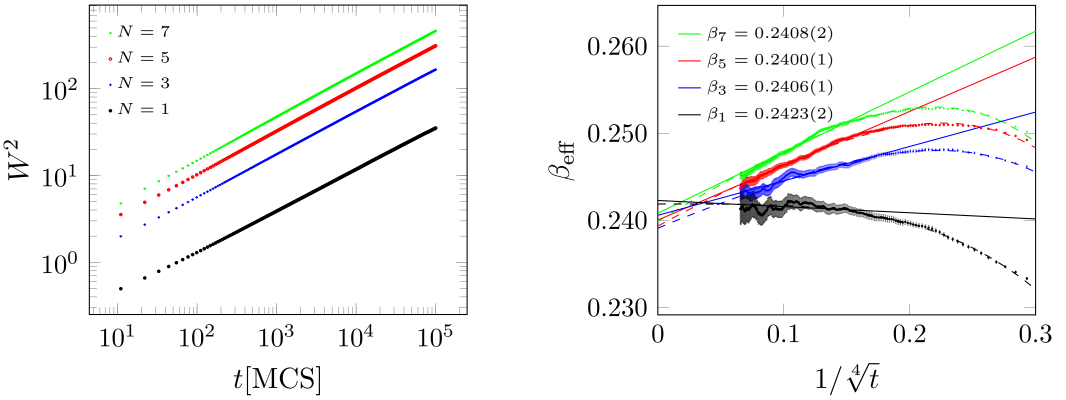


Figure 4.6.: Left: Squared roughness (W^2) of RSOS surfaces of size $V = 4096^2$ (256 realizations) in the scaling regime (error-bars are smaller than symbols). Right: Local slope analysis of roughness scaling of RSOS surfaces of size $V = 8192^2$ (128 realizations), straight lines are linear fits to the tail ($t \geq 1260\,\text{MCS}$), extrapolating to $t \to \infty$, assuming $\sqrt[4]{t}$ corrections. Error bars are propagated 1σ–errors, uncertainties given for β_N are pure fit errors. The black dashed line is the power-law extrapolation for $N = 1$. The dashed lines corresponding in color to the respective plots for $N > 1$ are fits of the form (4.13). All PL fits were performed for $t \geq 148\,\text{MCS}$. Both figures show $N = 1, 3, 5, 7$ (bottom to top).

12 GB on an NVIDIA K40 GPU with some memory to spare for RNG states. Results were averaged over $n = 768, 128$ and 128 realizations, respectively, where the latter two correspond to only one MS run.

Growth of the surface roughness follows apparently the same, clear, power-law for all considered N (figure 4.6, left). Local slope plots (figure 4.6, right) show an effective growth exponent $\beta_{\text{eff}} \approx 0.25$ for $N = 5, 7$ at $t \approx 1000\,\text{MCS}$ ($t^{-1/4} \approx 0.18$), which is in agreement with the results of Kim [83]. At later times the effective growth exponent decreases for all $N > 1$, which is followed over two orders of magnitude in figure 4.6. It can also be observed, that the maximum value β_{eff} increases with N. This suggests that the system is attracted to the fixed point of a random deposition model ($N \to \infty$), where $\beta_{\text{rnd. dep.}} = 1/2$ [81]. The behavior of the system is controlled by the finite value of N only at large-enough times.

Assuming independence of β from the restriction parameter N, it follows that the asymptotic estimates β_N should be the same for all N. Linear extrapolation to infinity, using corrections of the form $\sim \sqrt[4]{t}$ minimizes the variance of β_N for $N > 1$, which justifies this choice for the extrapolation. For completeness, it should be stated, that about the same minimal variance can be reached assuming logarithmic corrections of the form $\ln(t)^{2.2(3)}$, which also yields the same extrapolation results within the error margin. The power-law corrections allow better linear fits to the tails.

Table 4.2 lists the obtained estimates for β for the considered system sizes. Estimates for different $N > 1$ are practically identical and are thus averaged to give a common value. The case $N = 1$ is listed separately, due to the different corrections to scaling. For $N = 1$, β_{eff} can be best extrapolated by a power-law fit with exponent $x = 0.9(2)$. This is in good agreement with the results of [131], where $x \simeq 0.96 \simeq 4\beta$ is reported, based on the KPZ ansatz hypothesis. Since at the same time a PL tail with $x \simeq 0.25 \approx \beta$ was found for $N > 1$, this motivates testing PL corrections with exponents $x_i = m \cdot \beta$. Thus the two PLs found for $N = 1$ and $N > 1$ were combined into the form:

$$\beta_{\text{eff}}(1/t) = \beta + a_1/t^{4\beta} + a_2/t^{\beta} \quad , \tag{4.13}$$

where a_i are free parameters. The respective PL fits for all N were performed for $t \geq 148\,\text{MCS}$, showing good agreement with most of the growth region. See dashed lines in Fig. 4.6, right panel.

Effective scaling exponents β_{eff} in the case $N > 1$ exhibit corrections which are stronger and of different form than those for $N = 1$, as one can observe in figure 4.6. Furthermore, the present data suggests a possible oscillating convergence of β_{eff} for $N > 1$ as reported in reference [132] in simulations of the ballistic deposition model (BD). Extrapolations based on the form (4.13), while in good agreement with the observed region, are prone to over-fitting where they can not cover all present corrections. The values for $\beta_{N>1}$ are thus underestimated, if the effective exponents do indeed show oscillating convergence.

The estimates show no clear dependence on system size, thus it can be safely assumed that all simulations are well within the scaling regime and do not suffer from finite-size effects. All estimates agree with the previous estimate from the octahedron model $\beta = 0.2415(15)$ [27] within the margin of error. Most notably this is also the case for the estimates for $N > 1$. Overall, the presented data supports the estimate $\beta = 0.241(1)$.

Since all three curves in figure 4.6 (right) correspond to the same system size and have the same sample size, the figure shows, that the signal-to-noise ratio (S/N) in the simulation data increases with N. For $N = 7$, the S/N is lower by a factor of ~ 3.6 compared to $N = 1$ and by a factor of about ~ 2.5 compared with respect to $N = 3$. This relative decrease of noise may be caused by a kind of self-averaging, since systems with larger allowed height differences N accommodate more surface information at the same system size than smaller ones. It might pay to exploit this property by choosing larger N in studies of universal properties: While simulations for small N can be implemented more efficiently, a given level of accuracy can be

Table 4.2.: Estimates for the growth exponent β. Values in parenthesis are fit errors for $N = 1$ and 1σ error estimates for $N > 1$.

L	4096	8192	9605
β_1 (PL)	0.2412(1)	0.2418(1)	0.2415(1)
$\beta_{N>1}$ (lin.)	0.2404(3)	0.2405(3)	0.2410(3)
$\beta_{N>1}$ (4.13)	0.2395(3)	0.2394(3)	0.2399(2)

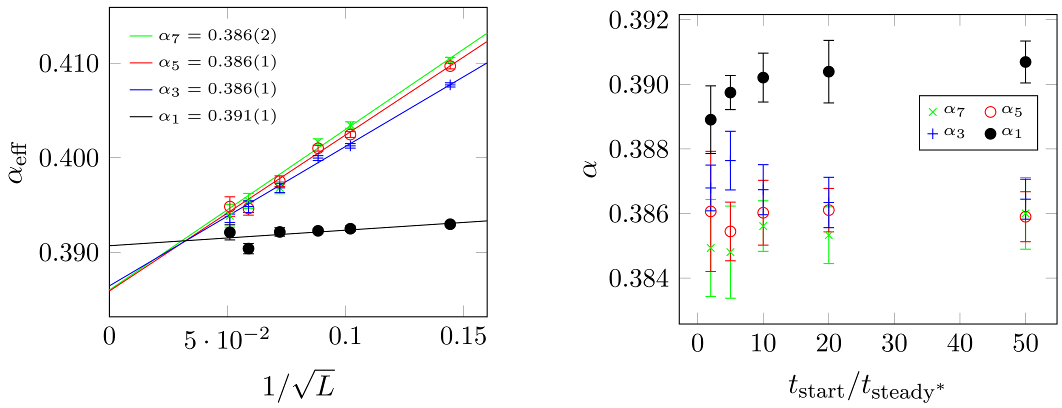


Figure 4.7.: Left: Local slopes of finite-size scaling analysis of RSOS with $N = 1, 3, 5, 7$. Error bars are propagated 1σ errors. Straight lines are linear fits to extrapolate to infinity, uncertainties given for α_N are pure fit errors. Steady-state data taken for $t > t_{\text{start}} = 50 t_{\text{steady}^*}$ (see text). Right: Dependence of extrapolated α on t_{start} is weak. Both figures: Sample sizes are at least 1024-2048 realizations and ≥ 8192 realizations for $L \leq 64$. All system sizes taken into account for finite-size scaling are listed in figure 4.8, where the considered timescales can also be read off.

reached with fewer runs using large N.

4.1.2.2. The Steady State

The roughness exponent α was determined using a finite-size scaling analysis taking into account the saturation roughness of system sizes between $L = 64$ and $L = 512$. In order to keep the noise amplitude constant, all calculations presented in the following used DD domain sizes of 8×8 lattice sites.

To determine the saturation roughness $W(L, t \to \infty)$ for each system size, values for $t \geq t_{\text{start}}$ of all available samples were averaged. In order to check whether the averaged values belong to the steady state and not to a transition phase, t_{start} was varied with respect to the apparent onset times of the steady-state t_{steady^*}, which was defined by the relation

$$a_N \cdot L^{\alpha} = b_N \cdot t^{\beta}_{\text{steady}^*} \tag{4.14}$$

This is just a rough estimate of the point where the power law of the growth phase reaches the average roughness in the steady state. The numerical parameters a_N and b_N where obtained by fitting to small systems.

Direct fitting of the scaling form

$$W_{\text{sat}}(L) \sim L^{\alpha}, \tag{4.15}$$

on the data for all simulated system sizes $32 \leq L \leq 512$ yields the following estimates for $t_{\text{start}} = 50t_{\text{steady}^*}$:

$$\alpha_{\text{fit}} = \begin{cases} 0.392(1) & 0.392(5) & \text{N=1} \\ 0.401(2) & 0.400(4) & \text{N=3} \\ 0.402(2) & 0.401(4) & \text{N=5} \\ 0.402(2) & & \text{N=7} \end{cases}$$

Here, the given errors are rounded-up fit errors. The second column shows Kim's estimates [83] for comparison. Values for all $t_{\text{start}} \geq 2t_{\text{steady}^*}$ fall inside the given error margins, although there is a slight increase in the estimates as t_{start} increases. These direct fits match perfectly the results obtained from sequential Monte Carlo simulations in [83].

However, if the first point at $L = 32$ is excluded, the estimates become significantly lower, pointing to strong corrections which can be seen in the effective exponents presented in figure 4.7. There is a clear tendency for α_{eff} to decrease with increasing system system size for $N > 1$. The approach to $L \to \infty$ is nonlinear but the number of points is insufficient for PL extrapolations to produce consistent estimates. In order to allow linear extrapolation, the data displayed in the plot has been linearized assuming a scaling variable $\sqrt{L}$. Extrapolation to asymptotically large systems then yields:

$$\alpha = \begin{cases} 0.391(1) & N = 1 \\ 0.386(1) & N > 1 \end{cases}$$

Corrections to finite-size scaling (4.15) at $N = 1$ are small, explaining the good agreement between local slope analysis and direct fit. The present data does not reach far into the steady state, which might cause a slight difference between the estimates for $N = 1$ and $N > 1$ as well as disagreement with a recent study [133] for $N = 1$, which found $\alpha = 0.3869(4)$ by performing the analysis at far later times. There is also further uncertainty in the extrapolation to $L \to \infty$ itself, which is not accounted for by the fit errors.

The observation of stronger corrections for larger allowed height differences N is consistent with a recent analysis of the BD. [132] The study found that corrections to scaling, for both α and β are reduced when the BD surface is smoothened by binning the surface positions before analysis, thereby decreasing the height differences between neighboring sites. Binning of the surface did not change the universal behavior, it only decreased non-universal corrections. The corrections produced an oscillatory approach to the asymptotic values of the exponents, which can explain why the extrapolations of α_{eff} (figure 4.7) and β_{eff} (figure 4.6) for $N > 1$ turn out lower than those for $N = 1$.

All presented estimates are in the range $\alpha = 0.389(4)$, which clearly excludes $\alpha = 2/5$ for $N \leq 7$.

Using the estimates for α and β obtained above, a good finite-size scaling collapse can be obtained for $N > 1$, even a perfectly looking one for $N = 1$, see figure 4.8,

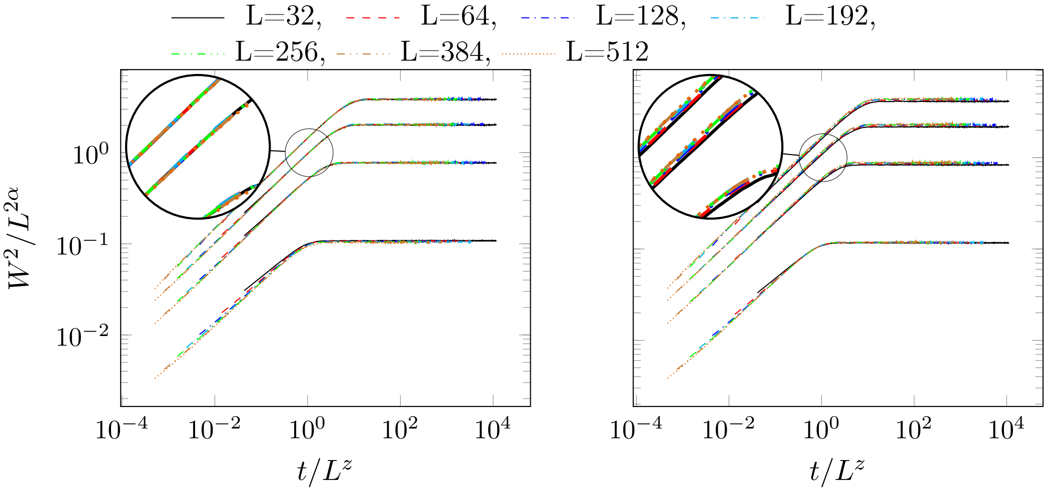


Figure 4.8.: Collapse of RSOS squared roughness into the steady state for $N = 1, 3, 5, 7$ (from bottom to top). The left figure shows a perfect collapse for $N > 1$, using $\alpha = 0.4$ and $\beta = 0.25$ ($z = \alpha/\beta = 1.6$). The right figure shows a collapse using $\alpha = 0.389$ and $\beta = 0.241$ ($z \approx 1.61$). The collapse looks perfect for $N = 1$ and good, but not perfect, for $N > 1$.

right. However, the collapse for $N > 1$ is imperfect for the growth phase whereas a perfect one can be achieved assuming the values suggested by Kim and Kosterlitz [71] (figure 4.8, left). The latter observation can be indicates the strong corrections to scaling (figure 4.6, right) and the corrections to the roughness exponent (figure 4.7, left): Effective exponents for early times and small systems do agree with the conjecture of [71] and indeed the mostly strongly outlying curves in figure 4.8, right, do belong to the smaller systems investigated.

To characterize the shape of the universal rescaled width and height distributions of the interface $P_L(\varphi)$, the standard measures skewness $S[\varphi]$ (4.6) and kurtosis $Q[\varphi]$ (4.7) have been calculated in the steady state.

The obtained values for the width-distribution $P_L(W^2(L))$ show no significant dependence on N nor L, our best results are $S = 1.70(1)$ and $Q = 5.38(4)$, in good agreement with those of [134].

For the distribution of surface heights, a weak correlation with the system size can be observed in Fig. 4.9. Heights were averaged in the steady state starting at different times $t_{\text{start}} > t_{\text{steady}*}$ (indicated by different symbols in the figure), but no dependence can be observed. Our results $S_h = 0.270(5)$ and $Q_h = 0.15(1)$ are in agreement with the ranges given in [125] and especially with the values $S_h = 0.26(1)$ and $Q_h = 0.134(15)$ reported in references [109, 126]. Thus, the cumulant values obtained for all N are in agreement with those published for the KPZ universality class, within the margin of error.

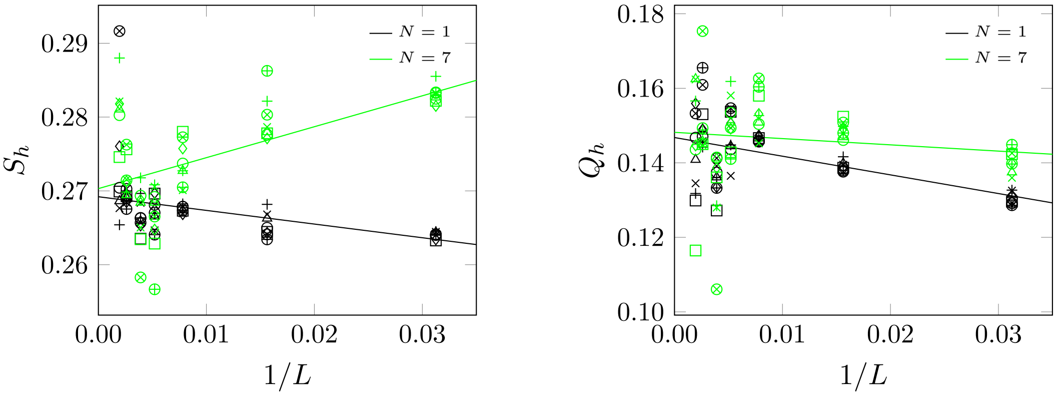


Figure 4.9.: Skewness S_h (left) and kurtosis Q_h (right) of the height distribution in the steady state plotted over the inverse lateral system size. Values are plotted only for $N = 1$ and $N = 7$ for the sake of clarity. The straight lines are linear fits, included to guide the eye. Different symbols indicate different ratios $t_{\text{start}}/t_{\text{steady}*} \geq 2$. A key is not provided for the symbols, because there is no correlation with this parameter.

4.1.2.3. Consistency of Fine-Size Scaling with Respect to DD

For comparison, additional finite-size scaling studies using DD domains of 16×16 and $6(+1) \times 10(+1)$ lattice sites were performed. The notation used to describe the latter indicates that, because the system cannot be divided into domains with a lateral size of six (or ten) lattice sites without remainder, a subset of domains have a larger lateral size to compensate, leading to an irregular tiling of the system. This configuration results from dividing the system into multiples of 5×3 tiles, in order to achieve optimal load balancing on NVIDIA GTX Titan Black GPUs. In both cases the smallest considered system is $L = 64$, because for smaller systems these domain sizes would become comparable to the system size. Another test was done using domains containing only $3(+1) \times 5(+1)$ lattices sites, which turned out to be too small to give correct results, as expressed by a failing data collapse. This decomposition is thus not considered in the following discussion.

Differences in the results between the considered DD configurations are not significant in data collapses nor in finite-size scaling fits. The most sensitive quantity is the effective roughness exponent, presented in figure 4.10. In these tests sample sizes are smaller than for the data presented above, making extrapolations less reliable. Still, all estimates derived from this data are consistent with the estimate $\alpha = 0.389(4)$ given above. Even the irregular, non-square, configuration does not deviate, although the steady state roughnesses might contain small systematic errors.

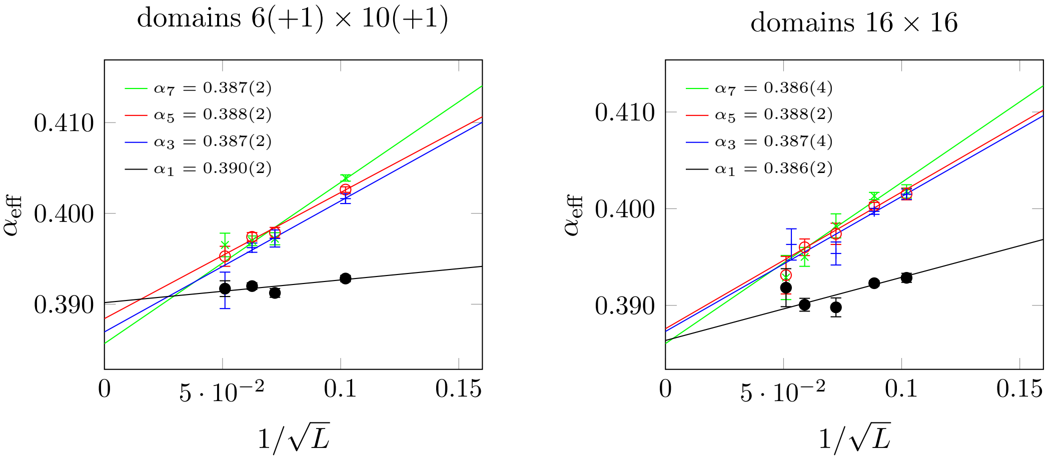


Figure 4.10.: Local slopes of finite-size scaling analysis of RSOS with $N = 1, 3, 5, 7$. Error bars are propagated 1σ errors. Straight lines are linear fits to extrapolate to infinity, uncertainties given for α_N are pure fit errors. Steady-state data is taken for $t > t_{\mathrm{start}} = 50 t_{\mathrm{steady}^*}$ (see text). Left: DD domains containing $6(+1) \times 10(+1)$ sites. Sample sizes are at least 512 realizations. For $N = 5, 7$ and sizes $L = 64$ and 128, 16384 respectively 8192 realizations are included. Right: DD domains containing 16×16 sites. For $L = 512$ the sample contains 256 realizations, for other system sizes at least 512 samples are included.

4.1.3. Results for Growth Phase and Steady State

Using extensive simulations of the octahedron model a new, more precise estimate for the scaling exponent of the KPZ universality class was obtained. Complementary simulations of the RSOS model for $N \leq 7$ have shown these estimates to also hold for $N > 1$, thereby providing evidence against the KK conjecture [71] also for $N > 1$.

The estimates for universal exponents are summarized in table 4.3. The best estimate for the scaling exponent is the one obtained from parallel RS simulations. A question mark is placed on the correctness of the β estimates under SCA dynamics at the obtained level of accuracy, where a weak dependency of β on p cannot be excluded numerically. They are thus not included in this discussion of the final results. However all estimates, including the SCA values, lie well within the margins of error of estimates found in recent literature [67, 135], and the earliest estimate based on GPU simulations $\beta = 0.2415(15)$ [27].

The estimate for the roughness exponent α obtained from RS simulations of the $N = 1$ RSOS model is in agreement with earlier estimates within $N = 1$ RSOS $\alpha = 0.393(3)$ by Parisi et. al. [109] and within the octahedron model [27] (listed in table 4.3). However, this value shows marginal disagreement with the more recent detailed study of finite-size scaling in $N = 1$ RSOS by Pagnani and Parisi [133] ($\alpha = 0.3869(4)$).

Inserting the best estimate for the scaling exponent $\beta = 0.2414(2)$ and the recent estimate by Pagnani and Parisi into the scaling law in equation (4.3) reveals that Galilean symmetry would be broken by more than twice the standard error. Since the studies presented in this work are focussed on the KPZ scaling regime, no precise estimates of the roughness exponent can be derived from a finite-size scaling analysis. The best prediction can be derived from the obtained value for β and the above scaling relation, yielding $\alpha = 0.3889(3)$. This is in agreement with the unified estimate from the RSOS study and marginally allowed according to the study in [27]. This indirect estimate for α claims the same level of accuracy as the most precise direct estimate $\alpha = 0.3869(4)$ [133]. These results disagree by five error margins, supporting a possible violation of the Galilei symmetry by KPZ, which was proposed to exist in by [136]. This prediction, regarding either α or a violation of the Galilean symmetry, remains to be tested directly by a more extensive finite-size scaling study using RS dynamics, because the study in [133] was based on SCA simulations. A MS code, as used for the presented RSOS simulations can proof an advantage in such an endeavour, but a study based on SCA simulations would also be interesting, if the influence of p was also quantified.

The best estimates from the present study are summarized in the bottom line of table 4.3, they are used in all further considerations.

Table 4.3.: Summary of estimates for KPZ universal exponents. The listed best estimate for α was calculated from the best estimate for β via the relation due to the Galilean symmetry.

Model	α	β	z	$\alpha + z \overset{!}{=} 2$
octahedron	[27] 0.393(4)	0.2414(2)	1.63(2)	2.02(3)
RSOS $N = 1$	0.391(2)	0.2415(3)	1.61(2)	2.01(2)
RSOS $N > 1$	0.386(2)	0.2407(6)	1.60(2)	1.99(2)
RSOS	0.389(4)	0.241(1)	1.61(3)	2.00(3)
RSOS $N = 1$	[133] 0.3869(4)	(0.2414(2))	1.603(3)	1.990(4)
best	0.3889(3)	0.2414(2)	1.611(3)	2

4.2. Autocorrelation Functions

The autocorrelation function is defined as:

$$C(t,s) = \langle \phi(t,\mathbf{r})\phi(s,\mathbf{r})\rangle - \langle \phi(t,\mathbf{r})\rangle \langle \phi(s,\mathbf{r})\rangle \tag{4.16}$$

$$\sim s^{-b}(t/s)^{-\lambda_{C,\mathrm{h,s}}/z} \quad , \tag{4.17}$$

where s is the waiting time, at which a snapshot is taken which is correlated with the system state at later times $t \geq s$. The function ϕ denotes the observable the autocorrelation of which is being measured. In this section, ϕ will take the form of the surface height $h(t,\mathbf{r})$ or the slopes, respectively the lattice gas variables of the octahedron model, $s(t,\mathbf{r})$. In case if $\phi \equiv h$, for $t = s$, the following holds:

$$\begin{aligned} C(s,s) &= \langle h(s,\mathbf{r})h(s,\mathbf{r})\rangle - \langle h(s,\mathbf{r})\rangle \langle h(s,\mathbf{r})\rangle \\ &= \left\langle h^2(s,\mathbf{r})\right\rangle - \langle h(s,\mathbf{r})\rangle^2 \\ &= W^2(L\to\infty, s) \sim s^{-b} \cdot f_C(1) \ . \end{aligned}$$

The latter proportionality points to the following relation, which must hold for the correlation function of heights at least at $L \to \infty$ and $s \to \infty$:

$$b = -2\beta \ , \tag{4.18}$$

with $\beta = 0.2414(2)$.

The autocorrelation exponent in $(1+1)$ dimensions was analytically shown to be $\lambda^{1\mathrm{d}}_{C,\mathrm{heights}} = 1$ [137, 138]. Later Kallabis and Krug conjectured, that this finding can be generalized as $\lambda_{C,\mathrm{heights}} = d$, where d is the dimension of the interface [139].

4.2.1. Comparison of DD Methods for RS Dynamics

4.2.1.1. Device-Layer DD

In the aging study published in [103] it was found, that the GPU implementation of the octahedron model with RS dynamics used therein exhibits an asymptotic autocorrelation function deviating from the sequential CPU reference implementation. This parallel implementation uses coarse dead border (cDB), where the DD origin is only moved on a coarse grid with 4×4 lattice-site units, see section 3.2.1, page 36. Figure 4.11 compares autocorrelation functions of both heights and slopes resulting when cDB is employed at device-layer with other schemes and sequential simulation results. For the slopes (right panel) one clearly observes convergence of the autocorrelation function to a finite value. The autocorrelation of heights under cDB can also be seen to deviate from the PL laid-out by the sequential reference. A finite asymptotic limit has not been reached in any of these simulations due to the short time-scale considered.

Intuitively, a finite asymptotic value for the autocorrelation function could mean

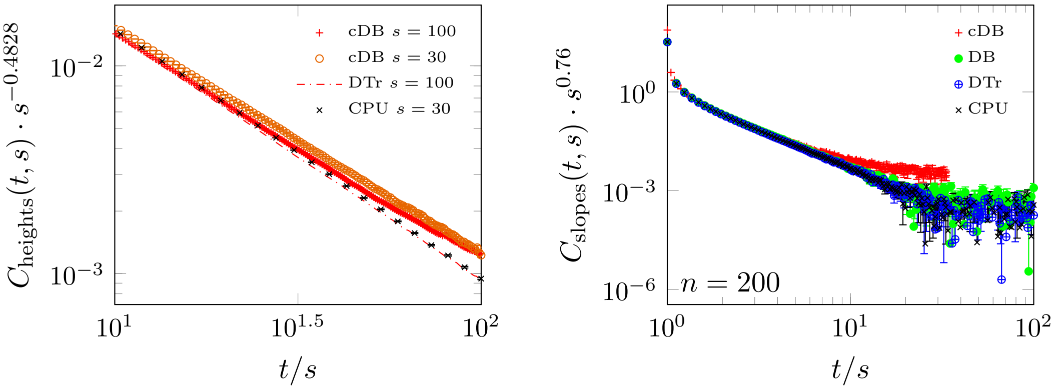


Figure 4.11.: Autocorrelations under RS dynamics with different device-layer DDs. Left: Autocorrelation of heights (data from [103]). The system sizes are $L = 2^{15}$, $L = 2^{16}$ and $L = 2^{13}$ in the cDB, the DTr and the sequential (CPU) runs, respectively. Sample sizes are: $n_{+} = 696$, $n_{\circ} = 830$, $n_{-\cdot-\cdot-} \geq 71$ and $n_{\times} = 4367$. Right: Autocorrelation of slopes for different types of device-layer DD compared to a sequential simulation at $L = 2^{13}$. All sample sizes are $n = 200$, so noise levels can be compared visually. The presented parallel runs use single-hit DT at block-layer.

that there is some pattern imprinted on the system during its evolution. When the DD origin is only shifted on a coarse grid, only lattice sites which lie on edges of this coarse grid can become borders. These sites are thus updated less frequently than the remaining sites, which never become border-sites. Hereby, two types sites are defined in the system evolving at different rates. Since these borders are only a single lattice site wide and device-layer domains are very large, the effect of this is apparently too small to be observed in the kinetics of surface roughening or steady state properties. It is, however, strong enough to imprint a persistent pattern onto the surface, which can be observed in the autocorrelation functions.

A straight-forward way to solve this problem, would be to not shift the DD origin on a coarse grid but allow arbitrary coordinates, which does indeed eliminate the observed correlations. For the present bit-coded implementation, unrestricted DB requires borders which are five inactive lattice sites wide. Moving the origin freely ensures that all lattice sites are updated with the same frequency, when sufficiently long times are considered. Ideally, border sites are only inactive for one asynchronous update sweep at a time (t_{async}). But, after moving the DD origin randomly, the old and the new borders will necessarily intersect at a grid of points, which are then inactive for $2t_{\text{async}}$. Due to the wide borders, these intersections cover patches of 5×5 lattice sites. Thus, the wide borders are producing locally varying update frequencies at short time scales. This manifests as additional noise in the autocorrelation functions and possibly other observables.

All curves presented in the right panel of figure 4.11 are averaged over the same number $n = 200$ of runs. At late times ($t/s > 30$) autocorrelation signals have decayed below the noise-level of the respective sample, so the variance can be compared. Reducing t_{async} decreases adverse effects of DD borders and thus also the additional noise caused by wide borders. The data shown for DB stems from simulation using $t_{\mathrm{async}} = 0.5\,\mathrm{MCS}$. Even larger noise is observed for $t_{\mathrm{async}} = 1\,\mathrm{MCS}$.

This study shows double tiling (DT) DD with random origin (DTr) to be superior to the other presented schemes: It appears to neither introduce correlation nor additional noise, compared with a sequential simulation. It has only the disadvantage of using smaller active domains for asynchronous update sweeps, but this influence is negligible for the large domain sizes usually used at device-layer and it serves to keep update frequencies homogeneous. For this reason it may even be preferable over DB with a border-width of one lattice site, since intersections of thin borders could still cause tiny imbalances. Using DT without randomly moving the DD origin results in similar correlations as exhibited by cDB.

All parallel RS surface growth simulations in this work, apart from examples presented above, used DTr at device-layer.

4.2.1.2. Block-Layer DD

Since at block-layer only single-hit updates are performed there is little potential for block-layer domain borders to have strong adverse effects on simulations results. However, since the type of block-layer DD does affect the corrections to scaling, as shown in section 4.1.1, an investigation of the effects on the autocorrelation functions in RS simulations shall be presented here.

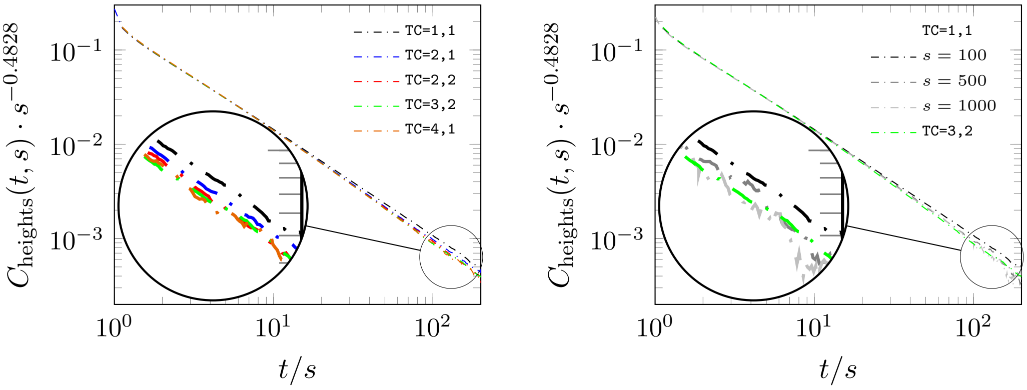
The observed changes in autocorrelation functions with block-layer DD are so small, that they could not be resolved by most sequential simulations. At the same time sequential and parallel simulations may differ by small corrections, not affecting any universal properties, which was already illustrated in section 4.1.1 as well. Thus, even if a small difference between parallel and sequential results could be resolved, it would be unclear if this was a cause for concern. For these reasons, comparisons with sequential simulations would not be constructive at this point.

In simulations using DD, the size and shape of domains remain free parameters, where the exact sequential behavior corresponds to the limit of infinite domain size. This view suggests checks for self-consistency as a viable method for this analysis.

Figure 4.12a shows autocorrelation functions for heights when using DTrDT for different block-layer domain configurations. Different TC configurations appear to show a trend in the autocorrelation functions at late times:

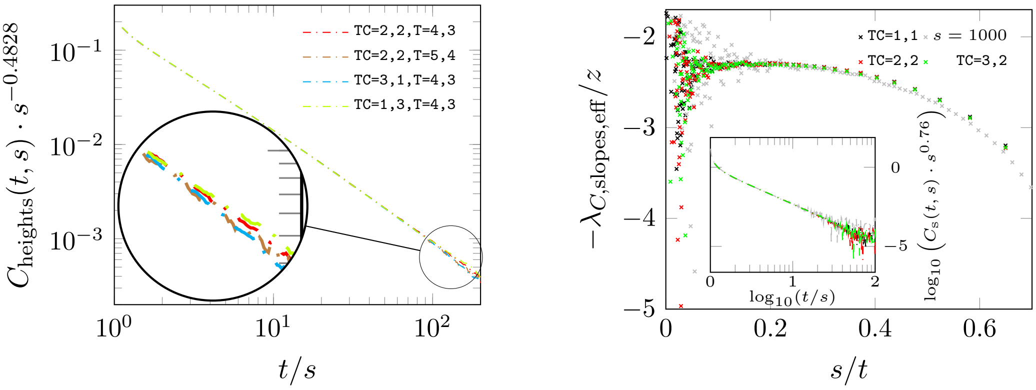
$$\texttt{TC=1,1} > \texttt{TC=2,1} > \texttt{TC=2,2} \gtrsim \texttt{TC=3,2} \approx \texttt{TC=4,1}$$

This trend does only depend on the volume of block-layer domains, but not on lateral dimensions, which is counter-intuitive, since if this effect arose from a correlation caused by the single-hit updates going into domains of the same set at the same time, some length-scale based on the smaller lateral domain size should dominate.



(a) AC of heights vs. TC (DTrDT)

(b) AC of heights vs. s (DTrDT)



(c) AC of heights (DTrDT)

(d) AC of slopes (DTrDT)

Figure 4.12.: This figure presents the effects of different TC configurations on RS autocorrelation (AC) results when using DT for DD at block-layer (DTrDT). (a): Comparison of different TC volumes and shapes for waiting time $s = 100$. (b): Comparison of different waiting times s for the smallest configuration `TC=1,1` vs. $s = 100$ for the largest considered configuration `TC=3,2`. (c): Comparison of different TC aspect ratios at fixed volume and different device-layer domain sizes for fixed TC for $s = 100$. The other panels do not list device-layer domain sizes (`T=x,y`–figures) because no dependence can be observed. (d) Comparison of autocorrelation (AC) of slopes for different TC configurations at $s = 100$ and, for the smallest configuration `TC=1,1`, $s = 1000$.
The system size for `TC=2,1` is $L = 2^{17}$, all other system sizes are $L = 2^{16}$. Sample sizes vary: $n_{\texttt{TC=1,1}} \geq 38; n_{\texttt{TC=2,1}} \geq 120; n_{\texttt{TC=2,2,T4,3}} \geq 71; n_{\texttt{TC=2,2,T5,4}} \geq 20; n_{\texttt{TC=3,1}} \gtrsim n_{\texttt{TC=1,3}} \geq 78; n_{\texttt{TC=3,2}} \geq 28$ and $n_{\texttt{TC=4,1}} \geq 30$. Statistical 1σ-errors are below 5×10^{-4} for all data points.

For the most part, the differences are barely significant, but a discrepancy between the configuration with the smallest volume block-layer domains, `TC=1,1`, and the rest can clearly be observed. If lateral cell dimensions dominated, the configuration `TC=4,1` would be expected to give more similar results, instead it much better agrees with `TC=3,2`, which features block-layer domains with the same volume ($V_{\texttt{TC=4,1}} = V_{\texttt{TC=3,2}} = 16 \cdot 2^5 = 256$ lattice sites).

The hypothesis of sole dependence on block-layer domain volume is supported by the comparisons in figure 4.12c: All presented curves are based on simulations with the same domain volume, but the aspect ratios vary, with no significant difference in the autocorrelation functions. The figure also shows two plots with the same TC configuration but different sizes of the device-layer domains. The lack of a significant difference between these curves (-·-·- and -·-·-) suggests that changing the size of device-layer domains does at least not change the autocorrelation functions enough to explain the difference observed in figure 4.12a.

Figure 4.12b shows results for `TC=1,1` for waiting times $s \in \{100, 500, 1000\}$ in comparison with results obtained using large domains `TC=3,2` at $s = 100$. Notably, in the `TC=1,1`-simulation, the autocorrelation function for $s = 100$ does not collapse with the other two $s/t \geq 30$. Instead, the latter agree with the displayed `TC=3,2`-simulation for $s = 100$, suggesting, that the observed influence of the block-layer domain size is largest at early times during the evolution, close to the flat initial condition. The stronger dependence on s for configurations with smaller thread cells, could point to some oscillation modulated onto the aging of the system by small thread cells. Under this assumption, the asymptotic exponent λ_c should still be the correct one even for small TC, but it would show for much later times or only at time scales longer than the length of the oscillation.

Contrary to the above observations for the autocorrelation of heights, the autocorrelation of slopes is identical in all these simulations. This is illustrated in figure 4.12d by overlaying the effective exponents $\lambda_{C,\mathrm{slopes,eff}}/z$, corresponding to the autocorrelation functions of the slopes shown in the inset.

It is probable that the disagreement on the exact form of the autocorrelation function between simulations with different block-layer domain sizes under DTrDT is only caused an by additional correction which becomes larger for small cells. Thus it it would be possible to extract the correct universal KPZ autocorrelation and aging exponents from any of the simulations compared here, asymptotically. Still, keeping corrections stable in parallel simulations would be advantageous for two reasons: First, as extrapolating the correct universal exponents with corrections of unknown form present is problematic, one should avoid introducing an additional, TC-dependent correction. Second, the complete form of the autocorrelation functions under RS dynamics may be of interest in later studies.

The reason DTrDT produces some small correlated noise may be found in the grid-like site-selection pattern at block-layer, where the coordinates for collective update attempts are restricted to the selected set of active domains, which resembles a grid. This grid-pattern cannot be eliminated by randomly moving the DD origin at the block-layer after each collective update (DTrDTr). In section 4.1.1 an observation

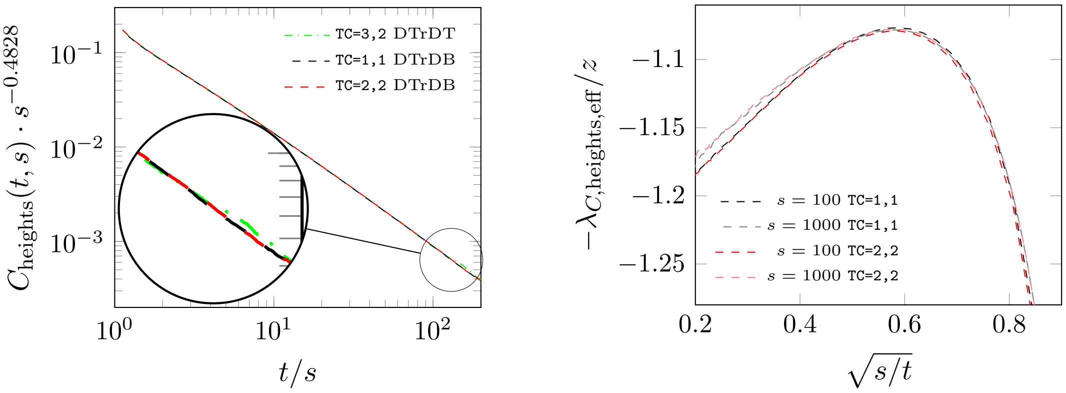


Figure 4.13.: Comparison of autocorrelation results in RS simulations using DB for DD at block-layer (DTrDB). Left: Autocorrelation functions for different TC sizes for waiting time $s = 100$. Results using DTrDT with TC configuration `TC=3,2` are given for comparison. Right: Local slope analysis of DTrDB results for $s = 100$ and 1000. All system sizes are $L = 2^{16}$. Samples sizes are: $n_{\texttt{TC=1,1}} \geq 1044; n_{\texttt{TC=2,2}} \geq 708$ and (DTrDT) $n_{\texttt{TC=3,2}} \geq 28$. Samples sizes of DTrDB runs are much larger because these stem from production runs which where used to extract final results.

is presented, that DTrDTr does not substantially improve on the results obtained using DTrDT, which is why it was not investigated in further detail.

The grid can be eliminated by turning back the DB scheme, in a single-hit variant, for DD of the block-layer (DTrDB). Figure 4.13 shows autocorrelation data for surface heights from DTrDB simulations using `TC=1,1` and `TC=2,2`. Here, the autocorrelation functions are in perfect agreement. The right panel shows local slope analyses for autocorrelation functions with waiting times $s = 100$ and 1000 from these simulations, which also agree almost perfectly. The data presented for DTrDB is taken from production runs, which are further analyzed in the next section. The curves are much smoother because of the larger sample size afforded.

The left panel of figure 4.13 also shows a curve from DTrDT simulations with `TC=3,2` for comparison. A good agreement cannot be denied, suggesting, that at this block-layer domain size, the DTrDT simulations are sufficiently converged with respect to the autocorrelation of heights. Since there is no significant dependence of the autocorrelation functions on the TC configuration when using DTrDB, the most efficient choice of DD for a simulation is DTrDB with `TC=1,1`.

4.2.2. Autocorrelation Properties under RS Dynamics

Final results for autocorrelation functions are displayed in figure 4.14. There is no significant difference in the scaling laws for different waiting times s, which makes a near-perfect collapse of the $C_{\text{heights}}(t,s)$ for different waiting times possible. The presented collapse (figure 4.14a) is achieved respecting relation (4.18). However, the best collapse is achieved for $b_{\text{heights}} = -0.469(3)$, which is caused by corrections to scaling being strong at early times, where $\beta_{\text{eff}}(t)$ is smaller than the asymptotic value (see figures 4.1 or 4.6) causing different aging.

The asymptotic tail of effective exponents are linearised assuming corrections of the form $\sqrt{s/t}$, as displayed in figure 4.14b. The exponent should in principle be independent of s, but corrections may differ, affecting extrapolation results. The extrapolated exponents are plotted over s in figure 4.14c. The plot appears to follow a rough trend for $s \to \infty$, but it does not lend itself for a clean fit. However, a PL-fit to the combined set of points displayed would give an extrapolated value for the exponent of $\lambda_{C,\text{heights},s\to\infty}/z \approx 1.21$. A weighted average of all points is dominated by small waiting times due to the smaller fit errors in the local slope extrapolations. Such an average would support an estimate of $\overline{\lambda}_{C,\text{heights}}/z = 1.256(15)$.

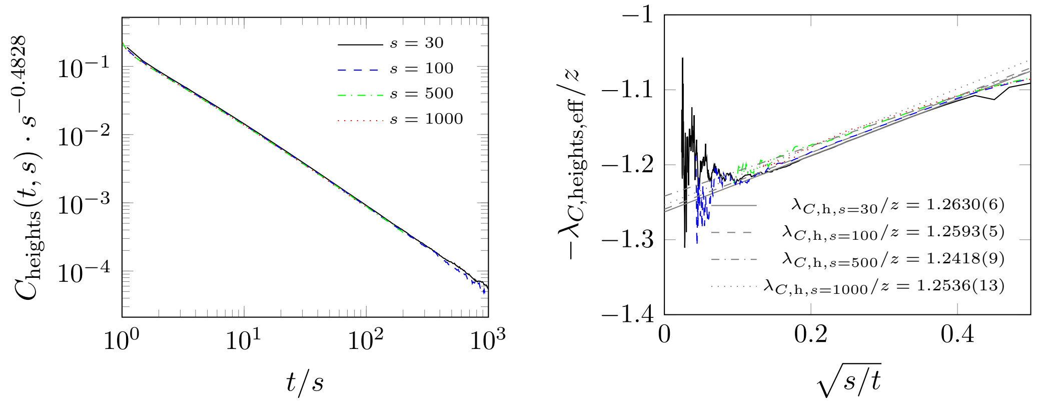
As an attempt to resolve this discrepancy, a different type of local slope analysis is presented in figure 4.14d, using tail effective exponents, as defined in section 2.1 (page 14). These effective exponents can be expected to converge more monotonically to the asymptotic exponent, because the tail of the autocorrelation function is included for all t_{min} and only increases in weight as t_{min} increases. In figure 4.14d the curves for different s approach the asymptotic exponents more linearly but with strong oscillations. However, all curves seem to oscillate around a common mean, which is not the case in figure 4.14b. A single linear fit to the combination of all curves, yields an averaged extrapolation of $\widetilde{\lambda}_{C,\text{heights}}/z = 1.23(3)$, suggesting, that the estimates presented before are under- and overestimates, respectively. The given error margin takes the uncertainty due to the actually unknown corrections into account. This yields a corresponding autocorrelation exponent $\lambda_{C,\text{heights}} = 1.98(5)$. Figure 4.14 mostly shows data for simulations using DTrDB, `TC=1,1`; the above estimates are also compatible with results obtained using `TC=2,2`; as evidenced by figure 4.14c.

The autocorrelation functions of the surface slopes in these systems are presented in figure 4.15. Here again, the functions for different waiting times collapse nearly perfectly, this time for $b_{\text{slopes}} = 0.76(2)$.

Since the autocorrelation functions for slopes decay much more rapidly than for heights, the signal-to-noise ratio in the present sample is insufficient for a reliable extrapolation of the asymptotic exponent from effective exponents. A weighted average of direct PL fits for $4 \leq t/s \leq 90$ yields $\lambda_{C,\text{slopes}}/z = 2.312(2)$. However, effective exponents show a downwards curvature as $t/s \to \infty$, suggesting an asymptotic value closer to $\lambda_{C,\text{slopes,eff}}/z = 2.39(2)$.

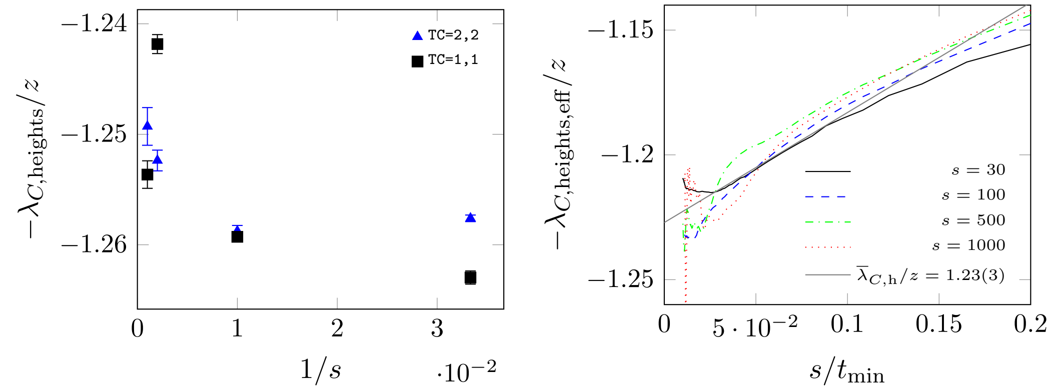
4.2.3. Autocorrelation Properties under SCA Dynamics

Stochastic cellular automaton (SCA) updates differ from RS updates in that they are spatially correlated, where the strength of the correlation is determined by the update probability $p < 1$. In the calculations presented here, p corresponds to the deposition probability, while the probability of removal q is set to zero, for the simulation to stay in the pure KPZ class. Figures 4.16 and 4.17 show autocorrelation functions for height variables and surface slopes, respectively, measured in SCA simulations at



(a) RS AC of heights

(b) AC effective exponents



(c) AC exponents

(d) AC tail effective exponents

Figure 4.14.: Autocorrelation results from RS calculations using DTrDB with `TC=1,1`. System size $L = 2^{16}$, $n \geq 1044$ realizations for $s > 30$ and $n \geq 473$ for $s = 30$. (a) Collapsed autocorrelation functions for waiting times $s = 30, 100, 500, 1000$. (b) Corresponding local slope analysis and extrapolations assuming corrections of the form $\sqrt{s/t}$, as drawn. Linear fit was performed for $\sqrt{s/t} \in [0.1, 0.3]$. Stated errors are pure fit-errors, see text for actual error margins. (c) Exponents $\lambda_{C,\text{h}}/z(s)$ as obtained in panel (b), corresponding values obtained in DTrDB `TC=2,2` simulations are displayed additionally (▲). (d) Tail effective exponents corresponding to panel (a) obtained from PL fits for intervals $t \geq t_{\text{min}}$ with successively increasing t_{min}. A linear fit to the combination of all curves is displayed (——).

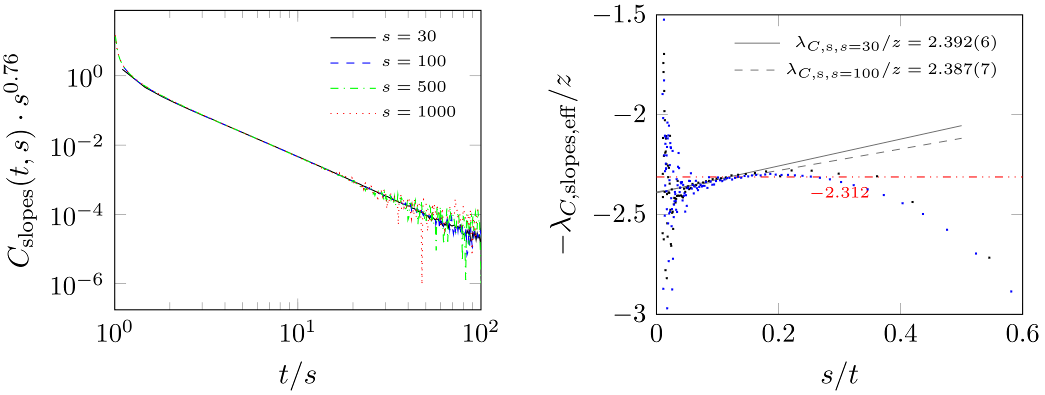


Figure 4.15.: Autocorrelation results from RS calculations using DTrDB with `TC=1,1`, same simulation as in figure 4.14. Left: Collapsed autocorrelation functions for waiting times $s = 30, 100, 500, 1000$. Right: Local slope analysis for $s = 30, 100$. Linear extrapolations are shown, assuming corrections of the form s/t, as drawn. The fit was applied in the interval $t/s \in [6.25, 50]$. The horizontal line (-··-) marks the value obtained from direct PL fits.

$p = 0.95$ as well as for $p = 0.5$.

The most apparent property of SCA autocorrelation functions, both of heights and slopes, is the finite asymptotic value (figures 4.16a and 4.17a, respectively). This limiting value, henceforth denoted by o, depends exponentially on p, thus it can be regarded as a direct manifestation of the correlation imprinted onto the surface by the SCA update pattern. The limit is independent of the system size and only shows a marginal dependence on the waiting time s manifesting in a slight decrease of o with increasing s. The cause of this remains unclear.

One exception can be seen in figure 4.17a for case ($p = 0.5, s = 30$), which exhibits a significantly lower limit than the other autocorrelation functions for $p = 0.5$. It can be hypothesized that this difference is caused by a perturbation introduced in the form of the flat initial condition, which at the waiting time s is superposed with the growth state of the KPZ surface. As such a perturbation can be expected to decay as

$$f_{\chi}(t) \sim t^{-\lambda_R/z}, \tag{4.19}$$

where λ_R denotes the autoresponse exponent, the shift of the autocorrelation limit would decay in the same fashion with s. More simulations for small s would be required to test this hypothesis.

The autocorrelation limit o for each function was determined using a linear extrapolation of the function's tail to infinity as a first approximation. Subtracting the appropriate limit from each curve reveals a PL approach to this constant. To obtain a refined value for o, the governing exponent ξ is read off the data, allowing

Table 4.4.: Autocorrelation limits for SCA dynamics for different deposition rates p and $q = 0$, as functions of the waiting time s. Given errors are fit errors, which are below the given number of digits for the slopes values. The value for $o_{\mathrm{slopes},p=0.95}(s \to \infty)$ is the steady state value, explained in section 4.2.4.

s/MCS	o_{heights}		o_{slopes}	
	$p = 0.5$	$p = 0.95$	$p = 0.5$	$p = 0.95$
30	0.003 20(3)	0.055 398(8)	0.012 871	0.221 623
100	0.003 31(5)	0.055 20(3)	0.014 286	0.219 827
500	0.0031(2)	0.054 57(8)	0.013 944	0.218 547
1000	0.0035(3)	0.0548(2)	0.013 903	0.218 330
$\to \infty$				0.218 00(5)

a subsequent fit of the tail of the form:

$$f(t) = o + c \cdot t^{-\xi}, \tag{4.20}$$

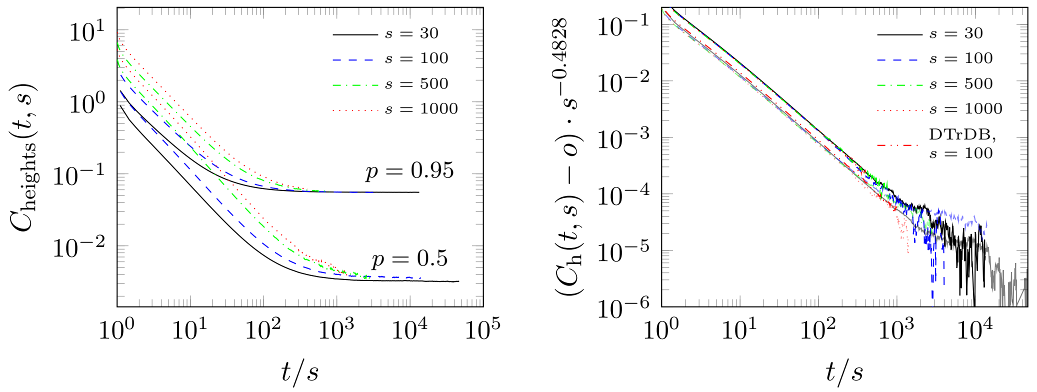
where o and c are free parameters. The corrected exponent $\xi' \to \lambda_C/z$ can then be read off after subtracting the refined o. These iterations yield self-consistent estimates for o and the autocorrelation exponent λ_C. This procedure is more prone to statistical error for small t/s since the considered function is further away from the asymptotic constant o, allowing noise in the tail to influence the extrapolated value more strongly. Table 4.4 lists calculated SCA autocorrelation limits.

Autocorrelation limits o for a range of p, could also be determined from the small survey study presented in figure 4.2, left, comprising much smaller sample sizes than the results presented in detail in the following. This data suggests an exponential dependence $o(p) \sim \exp(\nu p)$ with a similar, or possibly the same, value for the parameter ν for both slopes and heights. However, these autocorrelation measurements all used the same waiting time s, not taking into account p-dependent time-scale. Thus the actual waiting times $\widetilde{s}$ decrease with p, which makes the fit performed on the $o(p)$ across these runs unsuitable to determine a reliable value for ν.

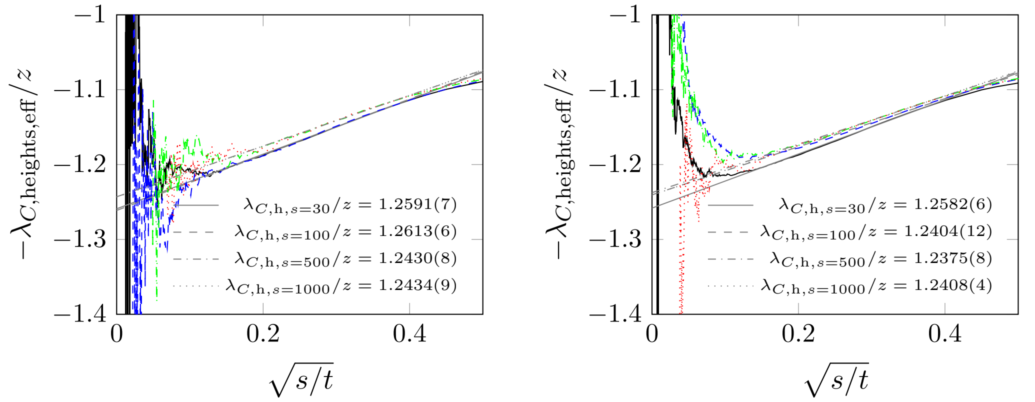
4.2.3.1. Autocorrelation of Heights

Figure 4.16b shows SCA autocorrelation functions for heights, corrected by subtracting the limiting autocorrelation o_{heights}. A nearly perfect data collapse can be achieved using the value same value for b_{heights} as determined for RS simulations. The RS autocorrelation function, displayed for comparison, shows behavior identical to its SCA counterparts.

Local slope analyses of the corrected autocorrelation functions are displayed in figures 4.16c and 4.16d for $p = 0.95$ and $p = 0.5$, respectively. Assuming corrections of the form $\sqrt{s/t}$, allows for a linear extrapolation of the asymptotic autocorrelation exponent. Only for $s = 30$ and 100 at $p = 0.95$ o can be determined reliably, the determined exponents for larger s may be influenced by an incorrect limit o. Thus,



(a) raw autocorrelation function (b) corrected autocorrelation function



(c) effective exponents $p = 0.95$ (d) effective exponents $p = 0.5$

Figure 4.16.: Autocorrelation of heights from SCA calculations. Error bars have been omitted for clarity. The visible noise is a good indication for 1σ error. Panels (a) and (b) show datasets with $p = 0.5$ (3062 realizations, $t \leq 1.4\,\text{MMCS}$) and $p = 0.95$ (3062 realizations, $t \leq 400\,\text{kMCS}$), curves from bottom to top. Lateral system size is $L = 2^{16}$. (a): Raw autocorrelation functions showing saturation depending on p. (b): Collapsed autocorrelation functions, corrected by the saturation offset o (see text). Plots for $p = 0.5$ and $p = 0.95$ corresponding to the same s use the same colors, where the bottom set of plots corresponds to $p = 0.5$. Colors are less saturated for the plots for $p = 0.5$, to distinguish them at late times. Data form a DTrDB run for $s = 100$ is displayed for comparison (-·-·-). The bottom panels (c) and (d), show the local slope analysis corresponding to the $p = 0.95$ and $p = 0.5$ data sets, respectively. Extrapolations assume corrections of the form $\sqrt{s/t}$, as drawn. Printed error margins are pure fit-errors.

only the extrapolations based on small s are considered for a weighted average. This yields the estimates $\lambda_{C,\mathrm{heights}}/z = 1.26(1)$ and $\lambda_{C,\mathrm{heights}} = 2.01(2)$ for the autocorrelation exponent. These values are in excellent agreement with the value obtained above from a local slope analysis of RS calculations for small s.

The increasing effective exponents at late times in figure 4.16d ($p = 0.5$) for $s = 30, 100$ and 500 are very unlikely to point towards a crossover into a different late-time regime. They are more likely random artifacts of the estimation procedure for o, caused by the low signal-to noise ratio at very late times. Ultimately the presented data constitutes strong evidence, that the autocorrelation of height variables in SCA simulations is identical to the one observed in RS calculations, up to constant o_{heights}.

4.2.3.2. Autocorrelation of Slopes

The picture turns out quite different for the slope variables. Figure 4.17b shows a working data collapse for both SCA data sets using the same value b_{slopes} as determined from RS calculations. However, the asymptotic PL approach to o_{heights} is quite different.

The dataset for $p = 0.95$ clearly exhibits a different exponent than is observed in RS simulations. Again, only the effective exponents (figure 4.17c) for $s = 30$ and 100 shall be considered, because of the better signal-to-noise ratio. A direct linear extrapolation for $s/t \to 0$ yields an estimate of $\lambda_{C,\mathrm{slopes,SCA}}/z = 0.75(2)$.

For $p = 0.5$, a transition from the power-law that is observed in RS simulations to the one determined above is clearly visible. Linear extrapolation of the tail in figure 4.17d yields $\lambda_{C,\mathrm{slopes,SCA}}/z$ (see figure), in good agreement with the corresponding RS results. This leads to the conclusion, that the autocorrelation function under SCA dynamics takes the form:

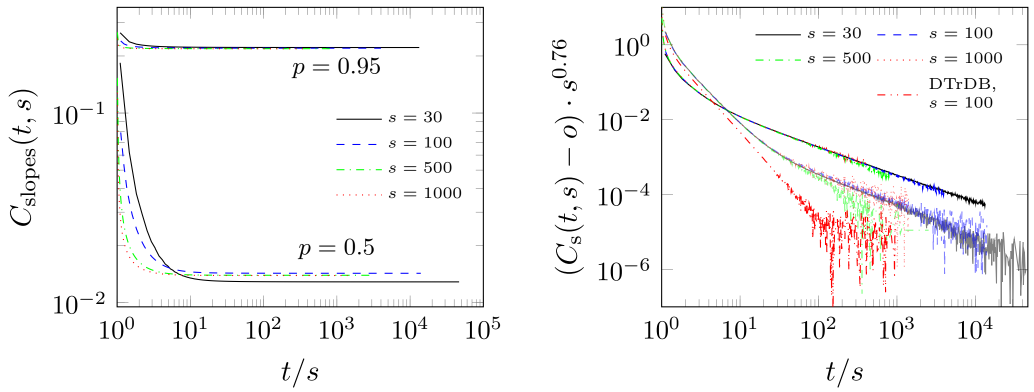
$$f_{C,\mathrm{SCA}}(t/s, p) \sim c_1 \cdot (t/s)^{-\lambda_{C,\mathrm{slopes}}/z} + c_2 \cdot (t/s)^{-\lambda_{C,\mathrm{slopes,SCA}}/z} \tag{4.21}$$

4.2.4. Autocorrelation in the SCA Steady State

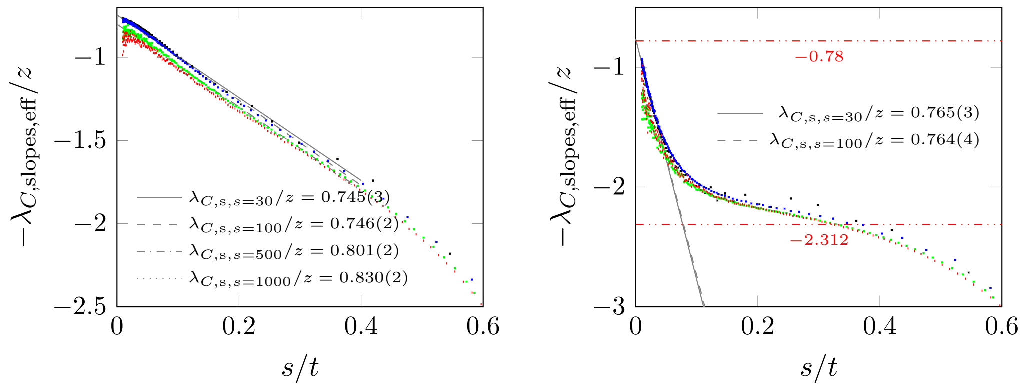
The autocorrelation functions in the steady state have not been explicitly investigated in this work, since no aging can be expected to be present after the growth regime. However, based on available SCA simulations in the steady state, some observations can be made.

Due to the broadening of the roughness distribution in the steady state with system size, no self-averaging of height-related variables is present for large systems. This can be observed to transfer to the autocorrelation of heights: Among the available datasets ($L = 2^{12}$ and $L = 2^{13}$, $p = 0.95$), the one for smaller systems exhibits the lower variance. However, for the autocorrelation of slopes perfect self-averaging appears to be present.

Because the simulations were designed to sample the roughness distribution over long times deep into the steady state, not to sample evolution at short timescales of $\Delta t \lesssim 1\,\mathrm{kMCS}$, the available data is not well suited to precisely determine the exact



(a) raw autocorrelation function (b) corrected autocorrelation function



(c) effective exponents $p = 0.95$ (d) effective exponents $p = 0.5$

Figure 4.17.: Results from SCA calculations for the autocorrelation of slopes. Error bars have been omitted for clarity. The visible noise is a good indication for 1σ error. Panels (a) and (b) show datasets with $p = 0.5$ and $p = 0.95$, curves from bottom to top. Data are taken from the same runs as in figure 4.16. (a): Raw autocorrelation functions showing saturation depending on p. (b): Collapsed autocorrelation functions, corrected by the saturation offset o (see text). Data form a DTrDB run for $s = 100$ is displayed for comparison (-·-·-). The bottom panels (c) and (d), show the local slope analysis corresponding to the $p = 0.95$ and $p = 0.5$ datasets, respectively. Extrapolations assume corrections of the form s/t, as drawn. Printed error margins are pure fit-errors. Horizontal lines (-·-·-) in panel (d) mark the asymptotic esponents for RS updates and SCA at $p = 0.95$, from bottom to top.

form of the tail. Apart from this, an oscillation around zero is to be expected for non-correlated (RS) updates. Since in SCA simulations a constant correction o is observed, the correlations should oscillate around the finite value o instead, if the correction was caused by an intrinsic correlation in the SCA dynamics.

Indeed, the autocorrelation of slopes is oscillating around $o_{\mathrm{slopes,steady}} = 0.218\,00(5)$. Assuming no aging in the steady state, this value should be identical to the limiting autocorrelation in the growth regime for $s \to \infty$ (listed as such in table 4.4). In line with equation (4.19), good agreement with the expression

$$o_{\mathrm{slopes}}(s) - o_{\mathrm{slopes,steady}} = c_o \cdot s^{-\lambda'_{R,\mathrm{slopes}}/z} \tag{4.22}$$

is found with $c_0 = 0.031(5)$ and $\lambda'_{R,\mathrm{slopes}}/z = 0.63(4) \approx 1/z$.

For a two–dimensional driven lattice gas with exclusion, a PL decay of the autocorrelation was found [140]. The slopes of the octahedron model also constitute a driven lattice gas; only the system is not driven along a lattice direction but along the diagonal and particles are only allowed to jump in pairs, giving rise to different exclusion rules. The available data is insufficient to clearly distinguish whether the tail is exponential or a PL.[2]

For the autocorrelation of heights, due to small sample size and the lack of self averaging in the present simulations, the limiting value can only be obtained as $o_{\mathrm{heights,steady}} = 0(5)$, which does not allow for any conclusions. However visual inspection of the tail suggests exponential decay rather than a PL, which would suggest a similar behavior as found in 1d [141–143].

4.2.5. Autocorrelation in the EW Case under SCA

4.2.5.1. Autocorrelation of Heights

As mentioned in section 2.4.1, the octahedron model falls in the universality class of the analytically solved EW equation, if $p = q > 0$. In $2 + 1$ dimensions, the autocorrelation function takes the form [144]:

$$C_{\mathrm{heights}} = c_0 \ln\left(\frac{t+s}{t-s}\right) \quad , \tag{4.23}$$

where c_0 is a model-dependent constant. This function approaches 0 for $t \gg s$ like a PL with exponent $\lambda_{C,\mathrm{h,EW}}/z_{\mathrm{EW}} = 1$, where $z_{\mathrm{EW}} = 2$.

In the previous sections it was shown, that the autocorrelation function $p > 0$, $q = 0$ (KPZ case) approached 0 asymptotically only in simulations with RS dynamics, while under SCA dynamics a finite value is approached asymptotically. When $p = q$ (EW case) the spatially correlated updates of the SCA do not affect the autocorrelation function for the heights in this way. Instead, equation (4.23) is reproduced also for late times, as illustrated in figure 4.18. The small deviation for very early

---

[2] An exponential fit $\exp(-\xi(t-s))$, yields $\xi = 0.000\,19(2)$ with a variance of residuals $V = 0.09$. A PL $(t-s)^{\zeta}$ yields $\zeta = 0.79(3)$ at $V = 0.04$. Both fits have 48 degrees of freedom.

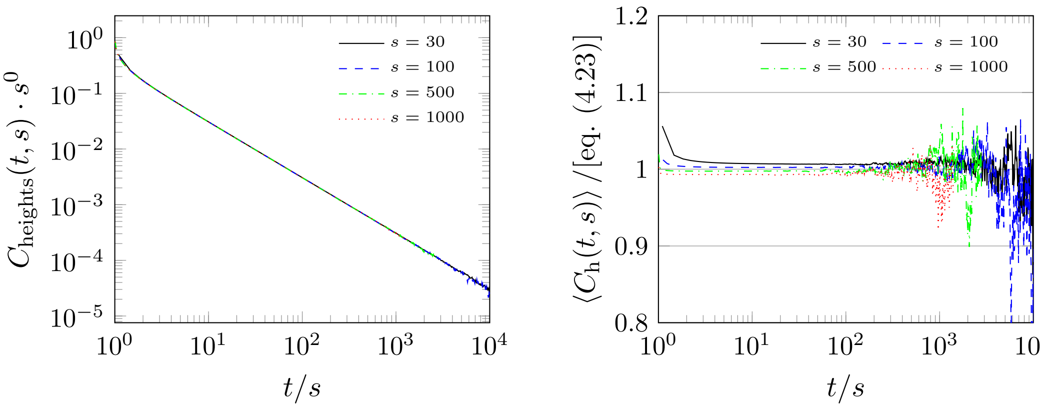


Figure 4.18.: Autocorrelation functions of heights under SCA dynamics with $p = 0.5$. Sample size is $n_{\mathrm{SCA}} = 5919$. Error bars are omitted for clarity. The magnitude of fluctuations can be seen from the visible fluctuations in the plots. Right: Data divided by fit of the exact form in equation (4.23) to the interval $10 < t/s$.

times can be observed in RS runs too and is most likely caused by a perturbation introduced by the flat initial conditions considered in the simulation.

The best fit values for c_0 for all waiting times s are ≈ 0.152. The fit values increase slightly with s. The exact value of c_0 for the octahedron model is likely to be found in the limit $s \to \infty$.

This does not only show the correctness of the SCA and RS implementations for the roughening kinetics down to a very small margin. The EW autocorrelation functions obtained from SCA provide an example of a system where correlations introduced by SCA do not affect the long time behavior.

Note, that, even though the aging exponent in the left panel of figure 4.18 is $b^{\mathrm{EW}}_{\mathrm{heights}} = 0$, physical aging is still taking place. This is indicated by the scale of the ordinate being t/s instead of $t - s$, the latter being the scale to be applied when a system is not aging, for example in a steady state.

4.2.5.2. Autocorrelations of Slopes

In RS simulations, the tail of $C_{\mathrm{slopes}}(t,s)$ does not decay with a single PL, as can be observed in the left panel of figure 4.19. The pronounced curvature in the log-log plot makes the form logarithmic. Assuming a logarithmic tail leads to the form $\sim \log^x(t/s)$, with $x \approx 4$. This is illustrated in the right panel of the figure, where the inverse of $C_{\mathrm{slopes}}(t,s)$ is plotted on a log-linear scale, where logarithmic decay appears linear. The exponent $x \approx 4$ is compensated by additional rescaling of the abscissa. However, the effective exponents (inset of the left panel) suggest a PL with exponent $\lambda^{\mathrm{EW}}_{C,\mathrm{slopes}}/z_{\mathrm{EW}} = 0.7(20)$. The two hypotheses about the form the tail under RS dynamics are tested using fits, which are presented in the right panel. The PL form appears to be better at describing the late-time portion of the data.

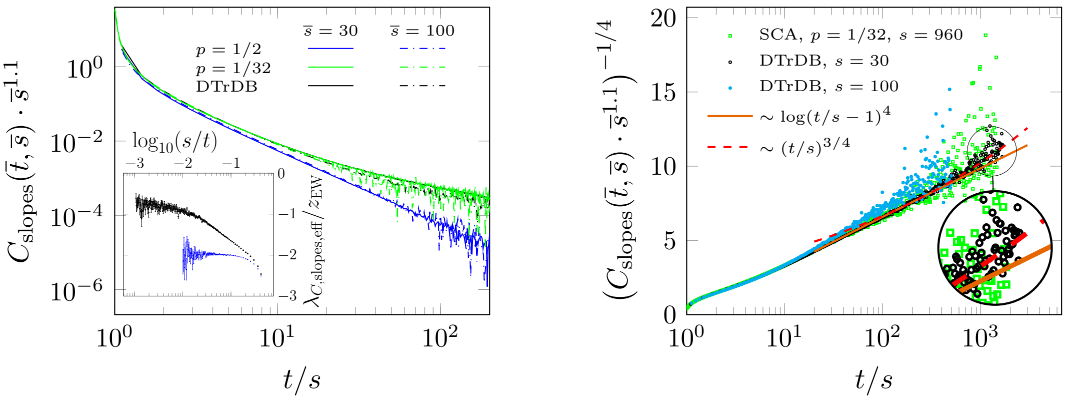


Figure 4.19.: Autocorrelation functions of slopes under SCA (—, —) and RS (—) dynamics. Sample sizes are $n_{—}=5919$, $n_{—}=147$ and $n_{—}=2101$, for both $\overline{s}=30$ and $\overline{s}=100$. For SCA, $p=1/32$, the simulation time is rescaled to collapse the curves, following: $\overline{t}=p\cdot t$, analogously for s. No rescaling is applied to the other plots: $\overline{t}=t$. Error bars are omitted for clarity. The magnitude of fluctuations can be seen from the visible fluctuations in the plots. The inset in the left panel shows the effective autocorrelation exponents $\lambda_{C,\text{slopes,eff}}/z_{\text{EW}}$ for DTrDB and SCA, $p=1/2$, both for $s=30$ and with 1σ error bars. Right: Plot of inverse data from RS and SCA, $p=1/32$, simulations on log-linear scale and to show the logarithmic tail. The abscissa is additionally rescaled by a power $1/4$ to linearise the tail according the log-tail hypothesis. See text for details. Two fits to the tail of the DTrDB dataset with $s=30$ are included: One follows the log-tail hypothesis, the other a PL with exponent $\lambda/z_{\text{EW}}=3/4$.

While SCA dynamics seems to perfectly reproduce the expected autocorrelation function for the surface heights, the evolution of the underlying lattice gas is changed. In the SCA simulations with $p=0.5$, which were presented in the previous section, the autocorrelation of slopes clearly exhibits a PL tail with $\lambda^{\text{SCA},0.5}_{C,\text{slopes}}/z_{\text{EW}}\approx 2$ (figure 4.19, left). In this case, no finite asymptotic correlation seems to be present, which is in contrast to the observations in the KPZ case, but which fits the assumption of correlations introduced by SCA site-selection cancelling in the EW case. The changed form of the tail may be caused by some part of the lattice gas kinetics, which slows decorrelation under RS dynamics, being suppressed by SCA dynamics.

In the limit where $p\to 0$ and $q\to 0$, SCA dynamics must necessarily approach RS dynamics. This is indeed the case here, as evidenced by data from SCA simulations at $p=1/32$ included in figure 4.19. The simulation time in these simulations is rescaled linearly with as $\overline{t}=p\cdot t$ to achieve the collapse with data from RS simulations, presented in the figure. This is an approximation to the rescaled time $\widetilde{t}$ defined in equation (4.10) for small p. In the RS case, the simulation time scales linearly with the value of $p=q\leq 1$. The observation of linear scaling under SCA dynamics with $p=1/32$ also suggests, that assuming SCA dynamics to have reached the RS limit at this value of p is a good approximation.

The aging exponent obtained from the presented simulations is $b^{\mathrm{EW}}_{\mathrm{slopes}} = 1.1(2)$. This value holds for both RS and SCA dynamics, but breaks down for very small values of s. The independence of the aging behavior from the dynamics is not surprising because the growth law of the surface heights, and thus the evolution of the system, remains unaffected, too.

These observations about the autocorrelation in this type of symmetric exclusion process of dimers in a two–dimensional lattice gas merit further investigation. Results for this system are not yet available in the literature. However, the EW universality class is not within the scope of this work. Within the scope of this work, these results provide an additional example of the manifold ways, in which stochastic lattice models can be affected by SCA site-selection dynamics, or remain unaffected in case of EW surface growth.

4.3. Autoresponse Functions

Autoresponse calculations were carried out by simulating two systems in parallel: An unperturbed system A and a perturbed system B. System A is simulated as a KPZ system with uniform deposition and detachment rates p_A and q_A, respectively. The perturbation in system B is introduced by simulating it with quenched disorder manifesting as lattice site-dependent rates $p_B(i)$ and $q_B(i)$ up to a waiting time s, after which $p_B = p_A$ and $q_B = q_A$.

The site dependent probabilities are implemented as a bimodal distribution, such each lattice site is assigned a pair of local probabilities $(p_B^{\pm}, q_B^{\pm})$. The disorder states $+$ and $-$ are equally distributed and the strength of the disorder is defined by a parameter ε, such that:

$$\begin{aligned} p_B^{\pm} &= \begin{cases} p_A \pm \varepsilon/2 & \text{if } p_B^{\pm} \in [0,1] \\ 1-\varepsilon/2 \pm \varepsilon/2 & \text{otherwise} \end{cases} \\ q_B^{\pm} &= p_A + q_A - p_B^{\pm} \end{aligned}$$

For the simulations presented in [103] (labeled CPU*) and in one SCA (labeled SCA*) simulation presented in the following section, the update probabilities were chosen such that $p_A + q_A = 1$ with $q_A > 0$ in order to guarantee more symmetric disorder. As long as q_A is small, the system is believed to remain in KPZ universality, when approaching $q \sim 0.5$ the system would cross over to EW universality.

The quantity of interest is the asymptotic behavior of the time-integrated response function:

$$\chi(t,s) = \int_0^s \mathrm{d}u R(t,u) = \frac{1}{L^2} \sum_i^{L^2} \left\langle \frac{\psi_i^B(t,s) - \psi_i^A(t,s)}{\varepsilon} \right\rangle = s^{-a} f_\chi(t/s) \quad , \tag{4.24}$$

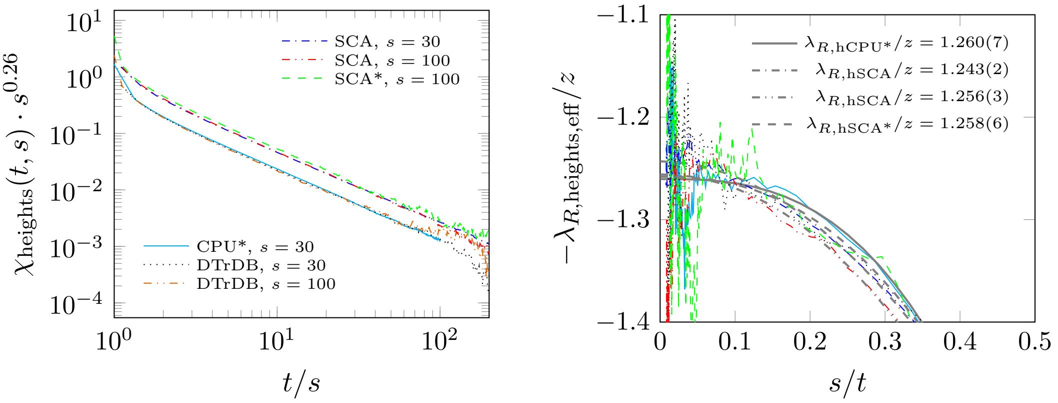


Figure 4.20.: Results for the autoresponse of heights, comparing variations of both RS and SCA simulations. Left: Collapsed autoresponse functions. Right: Corresponding local slope analyzes with PL fits for extrapolation of the asymptotic exponent. The extrapolation of DTrDB (-··-) was omitted because the late-time regime is too noisy. An extrapolation, which is in agreement with the remaining values can be obtained by fitting only up to $t/s = 20$ ($s/t = 0.05$). CPU* was published in [103]. System and Sample sizes are: $L_{\text{CPU}*} = 2^{13}$, $n_{\text{CPU}*} \geq 39083$ realizations, all others $L = 2^{16}$ with $n_{\text{DTrDB},s=30} \geq 830$, $n_{\text{DTrDB},s=100} \geq 629$, $n_{\text{SCA},s=30} \geq 23849$, $n_{\text{SCA},s=100} \geq 12012$ and $n_{\text{SCA}*,s=100} \geq 1390$ realizations.

where $\psi_i^{A,B}(t)$ are local observables of the unperturbed and perturbed systems, respectively. a is the aging exponent for the autoresponse, for which no relation to the other exponents of the $2+1$–dimensional KPZ class is known, whichis also due to the lack of an applicable FDR.

Asymptotically, the scaling function is expected to decay as $f_\chi(t/s) \sim (t/s)^{-\lambda_R/z}$. In order to obtain the response behaviour in the KPZ universality class, the height-functions $h_i^{A,B}(t)$ are used in these places and the exponent $\lambda_{R,\text{height}}/z$ can be read off. Inserting the field of slopes as $\psi_i^{A,B}(t)$ gives the autocorrelation properties of the underlying dimer lattice gas.

4.3.1. Autoresponse Properties

Results from this section and section 5.2 have been published as an article [145].

4.3.1.1. Autoresponse of Heights

Results from various autoresponse calculations are summarized in figure 4.20 (see figure caption for details). No difference in the autoresponse exponent and only a marginal difference in the corrections to effective exponents is evident, between random-sequential (RS) (CPU*, DTrDB) and stochastic cellular automaton (SCA) simulations. The most notable difference is a constant factor $(2.08(1))$[3] in the autoresponse function $\chi_{\mathrm{heights}}(s,t)$, which can be attributed to the rescaling of the time observed for SCA simulations in section 4.1.1 ($t_{\mathrm{SCA}} = \nu t_{\mathrm{RS}}$): While RS and SCA results do collapse using an aging PL s^a, as prescribed by the r.h.s. of equation (4.24), a prefactor ν^{-a} will remain between χ_{RS} and χ_{SCA}.

The value of the aging exponent is often determined by performing a manual collapse of the available datasets for different waiting times s. For RS simulations, the value $a_{\mathrm{RS}}^{\mathrm{coll.}} = 0.30(1)$ was determined this way and published in [103]. For the SCA simulations presented in figure 4.20, left, the value $a_{\mathrm{SCA}}^{\mathrm{coll.}} = 0.26(1)$ shows the best collapse. However, this method requires visual inspection of plots to determine for which value of $a^{\mathrm{coll.}}$ the data collapse works best, which is prone to bias and underestimation of the attached error margins.

Numerical computation of the aging exponent involves point-wise division of autocorrelation functions for different waiting times:

$$\frac{\chi_{\mathrm{h}}(t,s_1)}{\chi_{\mathrm{h}}(t,s_2)} = \frac{s_1^a f_{\chi_{\mathrm{h}}}(t/s_1)}{s_2^a f_{\chi_{\mathrm{h}}}(t/s_2)} \overset{(t/s_1=t/s_2)}{=} \left(\frac{s_1}{s_2}\right)^a$$

Numerical values $\langle\chi(t,s)\rangle$ are only available for discrete points t. Interpolation is required to compute these ratios at arbitrary t/s. The simplest option is linear interpolation, which can also be performed on a double-logarithmic scale, which reduces systematic interpolations errors when interpolating points following a PL. The present method yields $a_{\mathrm{SCA}} = 0.24(2)$, for the SCA simulations with $q = 0$, and $a_{\mathrm{DTrDB}} = 0.27(2)$, for our new RS simulations with $p = 1, q = 0$. For comparison, we calculated $a_{\mathrm{RS}} = 0.25(4)$ from the data published in [103], based on RS CPU and GPU simulations. From the present data, no significant difference between the aging exponents in RS and SCA simulations can be observed. The difference in β observed for theses types of dynamics found in section 4.1.1 does suggest a difference in the aging exponents, which, however, would likely be too small to be resolved by this aging study.

The near-identity of SCA and RS results is in strong contrast to the observations about the respective autocorrelation properties, but can be intuitively understood: For any autoresponse run, two simulations are performed which are identical up to a small perturbation by disorder in one simulation. When the autoresponse to the disorder is calculated, all other perturbations to KPZ–universality, which are present in both simulations, are bound to cancel. Due to this property of the response functions, SCA simulations can serve as a near perfect replacement for less efficient RS simulations when calculating response properties.

Table 4.5 lists the estimates for the autoresponse exponent $\lambda_{R,\mathrm{heights}}$. There is agreement across both the considered waiting times and RS and SCA dynamics.

A local slope analysis, based on tail effective exponents, as performed for the autocorrelation functions in section 4.2.2 does only work well for the dataset with the most samples (-·-·- SCA, $s = 30$). The tail effective exponents from the CPU*

---

[3] Calculated as the ratio of PL fits to the curves SCA($s = 100$) and CPU* in figure 4.20.

Table 4.5.: Estimates for the autoresponse of heights exponent $\lambda_{R,\mathrm{heights}}$, assuming $z = 1.611(3)$, obtained in section 4.1. Sample and system sizes are listed below figure 4.20. Error estimates are derived from fit errors and the error on z.

	CPU* [103]	SCA	SCA	SCA*
	$p = 0.98, q = 0.02$	$p = 0.95, q = 0$		$p = 0.95, q = 0.05$
	$s = 30$	$s = 30$	$s = 100$	$s = 100$
$\lambda_{R,\mathrm{h}}/z$	1.26(1)	1.25(1)	1.26(2)	1.26(2)
$\lambda_{R,\mathrm{h}}$	2.03(2)	2.00(2)	2.02(4)	2.03(4)
$\lambda_{R,\mathrm{h}}^{\mathrm{tail}}/z$	1.25(3)	1.23(2)		
$\lambda_{R,\mathrm{h}}^{\mathrm{tail}}$	2.01(5)	1.98(4)		

dataset exhibit strong fluctuations. The remaining datasets contain too much noise for consistent PL fits to the tails to be obtained. The exponents $\lambda_{R,\mathrm{h}}^{\mathrm{tail}}/z$ obtained from this analysis are also listed in table 4.5 do not agree with the ones obtained from the classical local slope analysis. This discrepancy cannot be resolved without knowing the corrections to scaling. To account for this uncertainty, it seems prudent to make a unified estimate: $\lambda_{R,\mathrm{h}}/z = 1.24(2)$.

4.3.1.2. Autoresponse of Slopes

The autoresponse signal of slopes, or lattice gas variables, is very weak and decays fast, hindering investigations based on the present data. Figure 4.21 shows the autoresponse function of slopes for two SCA datasets, the remaining datasets are too noisy to extract exponents and are thus not displayed. Based on the same argument made for the autoresponse of heights, there is likely no significant difference between autoresponse properties under SCA and under RS dynamics.

Even the SCA datasets contain too much noise to make a local slope analysis feasible. Exponents can be extracted by fitting a PL of the form $c_R \cdot (t/s)^{-\lambda_{R,\mathrm{s}}/z}$, yielding a unified estimate of $\lambda_{R,\mathrm{s}} = 4.5(2)$. Taking into account additional uncertainty from the choice of the considered interval as well as the fact that a direct PL fit would ignore possible corrections, a larger error margin should be assumed, even for a 1σ confidence interval.

4.3.1.3. Self-Averaging

To determine autoresponse properties large system sizes up to $L = 2^{16}$ have been used to exclude finite-size effects, like for all other simulations of the growth phase. Table 4.6 lists a figure of merit (FOM) for the S/N which is normalized by the computational effort due to system size (see table caption for details). A low FOM indicates a high S/N. Values being equal across different system sizes indicate the presence of perfect self-averaging in larger systems, thus using larger systems for autoresponse calculations effectively comes at no additional cost.

The FOM for SCA simulations is larger than that for RS simulations by a factor of about 1.2. Assuming a Gaussian distribution, one can estimate that SCA samples need to be about 1.44 times larger to reach the same statistical significance as RS samples. For the SCA simulations presented in this section, the local implementation of SCA for the octahedron model presented in section 3.3.1 was used, which is about 1.5 times faster than DTrDB. A speedup could be achieved by using the non-local implementation detailed in section 3.3.2. Thus, SCA dynamics can serve to vastly speed up the calculation of autoresponse functions in surface growth models.

It remains unclear why the CPU* runs from reference [103] show a particularly low level of noise. The same S/N is not reproduced by smaller test samples using the current version of the code. The only apparently relevant change was fixing a normalization error in the computation of the autoresponse functions, which should not affect the S/N. Reproducing the old results was not possible within the scope of this work, since obtaining an equally large sample size is computationally very expensive.

4.4. Summary

Since SCA had originally been employed only for bit-vectorized implementations, the update probabilities $p = 0.5, q = 0$ also were the natural choice and no study on the dependence of results on p were published. Such dependencies turned out rather subtle, except for the case of the autocorrelation of slopes. The extensive simulations of the octahedron model under SCA dynamics presented here, provide the first detailed study of surface growth and aging under this type of dynamics, which is accompanied by sufficiently precise data for RS dynamics for comparison. It is conclusively shown, that the autocorrelation of slopes is governed by two separate PLs under SCA dynamics, one of which is the PL present under RS dynamics, where

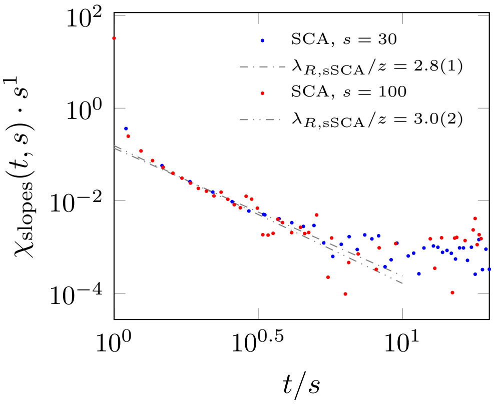


Figure 4.21.: Collapsed autoresponse functions of slopes from SCA simulations. Data extracted from datasets listed in figure 4.20. Given errors are fit errors.

Table 4.6.: Figure of merit for the signal to noise-ratio in the autoresponse-of-heights data. The value is based on the sample standard deviation multiplied by the lateral system size to show the effect of self-averaging due to system size. The values are constant in time, thus are shown averaged over time and the standard deviation of the series is given in brackets as a confidence interval for the variance. The values for SCA are additionally divided by the ratio of SCA and RS autoresponse signals $(2.08(1))^3$ to improve comparability.

L	SCA		CPU* [103]	CPU		DTrDB
	$s = 30$	$s = 100$	$s = 30$	$s = 30$	$s = 100$	$s = 100$
2^{12}	178(5)		64(2)	146(8)	262(16)	
2^{13}			64(2)			
2^{16}	175(4)	278(7)				231(19)

the crossover between these two depends on p.

The asymptotic autocorelation functions for the slopes of an EW surface were also shown to be different for RS and SCA dynamics. Under RS dynamics, strong corrections at early times, leading up to an asymptic PL with $\lambda^{\mathrm{EW}}_{c,\mathrm{slopes}}/z \approx 0.7$ have been calculated for the first time. Finite-time corrections under SCA danymics are smaller, while an asymptotic PL with the exponent $\lambda^{\mathrm{SCA},0.5}_{C,\mathrm{slopes}}/z_{\mathrm{EW}} \approx 2$ was determined. Contrary to the observations in KPZ case, SCA dynamics does not lead to finite asymptotic value here, neigher for the slopes nor for the heights. This causes the form of the autocorrelation function of heights to agree the analytical solution for the EW universality class.

Table 4.7 summarises the obtained autocorrelation and autoresponse exponents for the KPZ universality class, and the modified values under SCA dynamics. The autocorrelation exponents of heights agree with the conjecture $\lambda_C = d(= 2)$ by Kallabis and Krug [139] within the assumed margin of error, for both RS and SCA dynamics. The computed estimates for λ/z are given for reference. It should be noted, that using the estimate for λ_C published in [27] this would be marginally excluded.

The present results suggest for the relation $\lambda_C = \lambda_R$ to hold. This would be a nec-

Table 4.7.: Summary of KPZ autocorrelation and -response exponents. Assuming $z = 1.611(3)$. Estimates for SCA autocorrelation exponents include only small s.

	$\lambda_{C,\mathrm{heights}}$	$\lambda_{C,\mathrm{slopes}}$	$\lambda_{R,\mathrm{heights}}$	$\lambda_{R,\mathrm{slopes}}$
RS	1.98(5)	3.8(2)	2.00(5)	N/A
SCA	2.01(2)	1.25(2)	1.98(4)	4.5(4)
	$\lambda_{C,\mathrm{heights}}/z$	$\lambda_{C,\mathrm{slopes}}/z$	$\lambda_{R,\mathrm{heights}}/z$	$\lambda_{R,\mathrm{slopes}}/z$
RS	1.23(3)	2.35(10)	1.24(3)	N/A
SCA	1.26(1)	0.78(4)	1.24(2)	2.8(2)

Table 4.8.: Summary of KPZ aging exponents. Blank cells indicate identical predictions for RS and SCA.

	b_{heights}	b_{slopes}	a_{heights}
RS	$-0.4828(4)$	$0.76(2)$	$0.27(3)$
SCA			$0.24(2)$

essary condition for a FDR to hold in this system. Of course for out-of-equilibrium problems, such as surface growth, the existence of a FDR is not necessarily to be expected. Table 4.8 lists aging exponents obtained for autocorrelation and autoresponse from data collapses over s. They are not equal. Furthermore the relation between aging exponents, found in $1d$, equation (4.2), does clearly not apply here:

$$\begin{aligned} 1 + a &= 2\,(\beta + \beta/\alpha) \\ 1.30(1) &\neq 1.724(3) \end{aligned}$$

While this shows, that the FDR (4.1) does not hold in (2+1) dimensions, the equality of λ_C and λ_R hints at the possible existence of another generalized version, which may yet be found.

Based on the data primarily generated for aging studies, estimates for the scaling exponent β can be made to unprecedented accuracy, the final results being summarized in table 4.3. Here too, the accuracy of the present SCA simulations is large enough to pose the question, if surface growth under SCA dynamics deviate subtly from KPZ scaling universality.

However this question will be answered, it was shown that SCA scaling exponents are identical to KPZ at least up to three digits and that there is very good agreement in the steady state distribution (figure 4.5). This may allow using SCA dynamics for example to search for an upper critical dimension, which is advantageous, because especially in higher dimensions, the SCA algorithm would provide a large performance advantage over RS updates.

5. Further Topics

This work has been in part about the development of fast and efficient MC codes on GPUs and analysis the statistical properties of which. In so far it is supposed to provide possibilities to investigate further problems in non-equilibrium systems. Two topics which were only explored briefly and which can be investigated using the programs presented in chapter 3, shall be briefly presented in this chapter together with preliminary results.

5.1. Investigations of the Potts Model

GPUs are best suited for problems which are computationally dense, where all elements of the solution need to be computed explicitly or all MC updates must be attempted. Optimizations taking advantage of the sparsity of a problem can often not be implemented in any efficient way on GPUs, or sometimes even other parallel architectures. An example for this are continuous-time MC methods [94–96, 98], which require a large amount of bookkeeping for each sequential update to be performed.

In Ising or Potts models, the early stages of phase separation present a dense problem, where optimisations trying to make use of sparsity are ineffective. Systems which additionally contain quenched disorder, may relax too slowly for sufficient sparsity to occur even at late times (see section 3.4.2). This is why another application of the SkyMC code (section 3.4), apart from studies of RSOS models presented in section 4.1.2, is the study of domain growth specifically in disordered Potts models. This outlook is presented in this section.

5.1.1. Testing Results from the Parallel Implementations

It is clear, that Metropolis updates are far less sensitive to correlations introduced by DD than MC updates in surface growth models. Nevertheless, some evidence of the correctness of the parallel implementation of the Metropolis algorithm for non-conservative Potts models shall be presented. Figure 5.1 shows plots of the average domain size R growing during coarsening after a quench from infinite temperature in non-conserved Potts models on a square lattice for different lateral systems sizes and q. Here, the average domain size is defined as the average length of domains in both x and y direction [146].

The results of the GPU implementation and a sequential CPU implementations are identical. All simulations show PL growth with an exponent $1/2$ in the growth phase between the initial quench and the finite-size region, commensurate with the

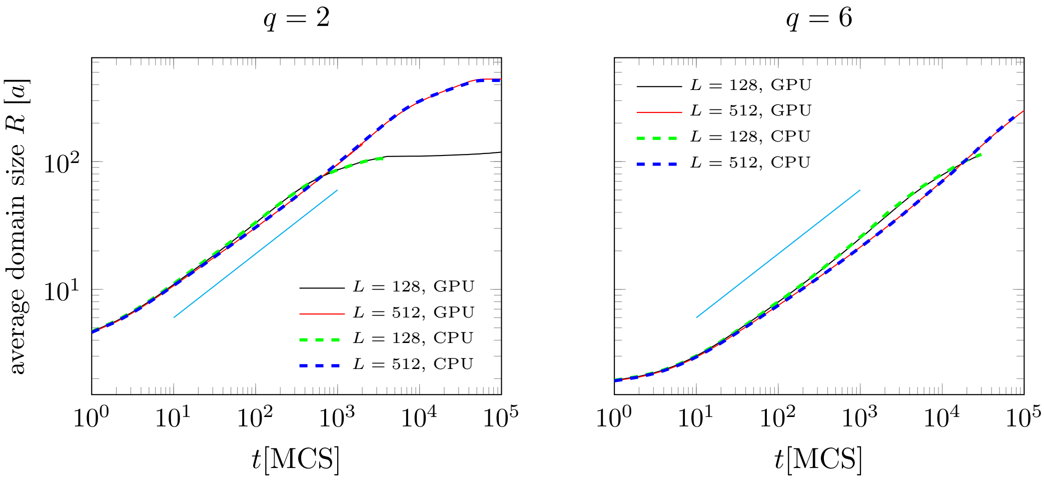


Figure 5.1.: Average domain size R in pure Potts models over simulation time in GPU (solid) and CPU (dashed) simulations. Simulations cells contain a square lattice with lateral sizes $L = 128$ and 512. Effective temperature is $K = 1.67K_c$. Left: $q = 2$ (Ising), Right: $q = 6$. Error bars are not shown, because they would be less than two line-widths in size. The solid, cyan lines represent PLs $\sim t^{1/2}$.

Lifshitz–Cahn–Allen law [47, 48], which is expected for spinodal decomposition in non-conservative systems.

In the presented case of the non-conserved Potts model, the average domain size reaches the systems size in equilibrium. The averaged curves in figure 5.1 show very slow, apparently logarithmic, growth just before reaching the maximum size. However, in this regime most of averaged realizations have already equilibrated. A few are frozen into meta-stable stripe patterns [147] which decay very slowly, causing the logarithmic growth of the ensemble average.

Direct comparisons of this type become more time-consuming with Kawasaki dynamics and in systems with quenched disorder because the coarsening kinetics is much slower in such systems. However, short simulations have shown no differences between GPU and CPU results at least for early stages of coarsening. Only rough tests of the critical temperature have been performed. It has been shown earlier, that DD, even a more correlated type than used in the present code, does not alter the critical temperature of Ising systems [93].

5.1.2. Domain Growth in Disordered Potts Models

The main goal of future investigations of disordered Potts models could be deciding the question about super-universality (SU), which was introduced at the end of section 2.3.2. Ultimately, this question revolves around the type of the asymptotic growth law in disordered systems. It is clear that the growth slows down with stronger disorder, leading at least to a smaller growth exponent [148, 149]. It remains unclear whether, or under which conditions, the PL form holds asymptotically, with an exponent depending on temperature and disorder parameters [56], or the growth crosses over to a logarithmic growth law asymptotically [58].

To distinguish numerically between a PL with a small exponent and logarithmic growth is hard, because this distinction can only be made when the growth is followed of multiple orders of magnitude in both time and domain size. Even in the case of an asymptotic PL, thus a finite asymptotic growth exponent θ, the effective exponents θ_{eff} may approach θ slowly, necessitating extrapolation of $\theta_{\text{eff}}(t)$ to $t \to \infty$.

Some simulations of the systems with bond-dilution have been performed. In such systems the strength of the disorder is characterized by the fraction of broken bonds $d < 0.5$, which must be below the percolation threshold for the system to stay ferromagnetic. To be able to compare effects of changing disorder and reduce temperature effects, one can choose simulations temperatures for different disorder strengths relative to the mean-field critical temperature, equation (2.11) $K_{c,\text{MF}}(d)$.

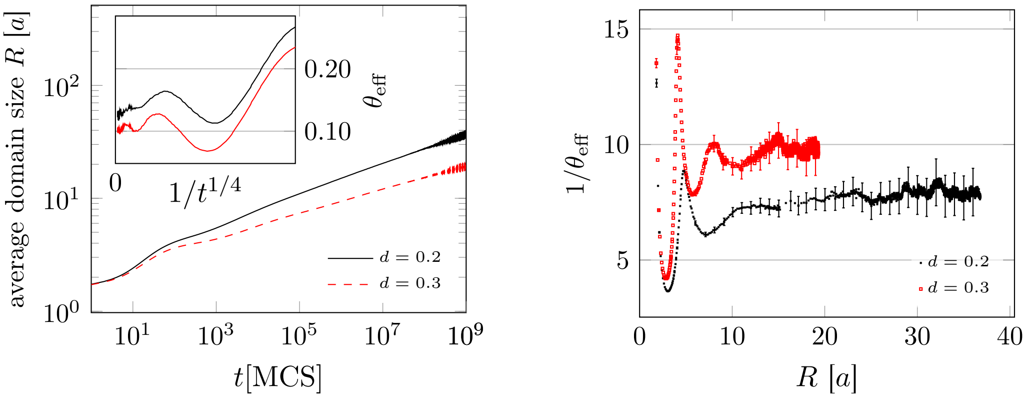
Results of very long runs for bond-diluted Potts models are shown in figure 5.2. For the case $q = 8$, the inset in panel 5.2a suggests an asymptotically finite exponent θ for both investigated disorder strength, this exponent decreases as d increases. The apparent lack of a crossover to a logarithmic growth law in this systems seems to supports the SU hypothesis. On the other hand, in the case $q = 2$, $\theta_{\text{eff}}(t)$ in panel 5.2c show a super-linear decay as $t \to \infty$, which could indicate a crossover to logarithmic growth, asymptotically. Comparing the doubly logarithmic plots of the domain size, it is apparent, that the curve for $q = 2$ (figure 5.2c) is distinctly less straight, suggesting slower-than-PL growth, thereby contradicting SU for this model.

The $1/t$ axes in the θ_{eff}-plots (figures 5.2a and 5.2c) are rescaled by an exponent $1/4$ to stretch the late time region of the plot. This rescaling does also linearize the tail for $q = 8$ to some extent, pointing to a possible leading order correction with a similar exponent. Another way to compensate correction to the growth law has been suggested in reference [148]: Plotting $1/\theta_{\text{eff}}$ against the domain size. This is shown, correspondingly, in figures 5.2b and 5.2d, where after strong fluctuations during the initial stages of domain formation and coarsening, θ_{eff} decays rather linearly. For $q = 8$, $1/\theta_{\text{eff}}$ appears almost constant, possibly indicating convergence to a finite value. No sings of convergence can be observed for $q = 2$. This is no proof of logarithmic growth, however.

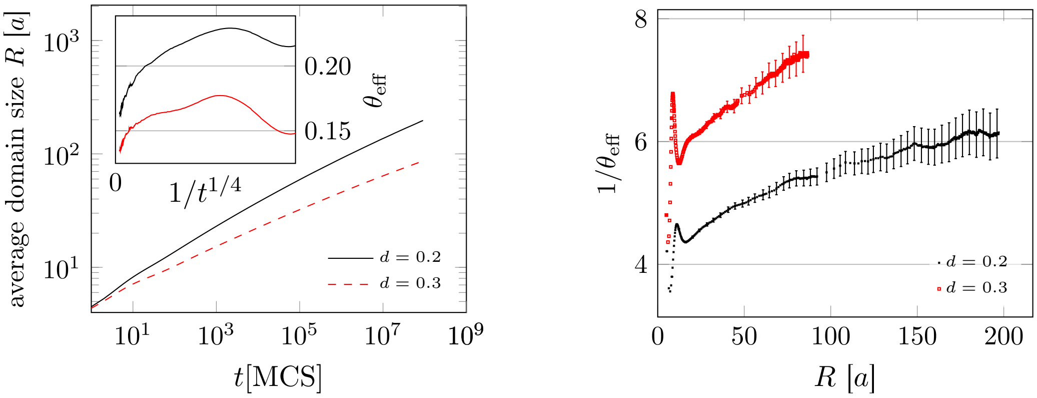
The function $1/\theta_{\text{eff}}(t)$ in figure 5.2d, does not appear to feature any discontinuities or kinks in late stages of coarsening. This suggests, that the growth laws presented for $q = 2$ do no suffer from finite-size effects, thereby excluding them as the cause of the steep decay of θ_{eff} at late times which is observed in figure 5.2c.

These two examples already show contradicting results regarding SU, prompting the immediate question, whether the presented $q = 2$ case does indeed cross-over to logarithmic growth. If this is the case, the next question may be whether in the case $q = 2$ broken-bond disorder is affecting the equilibrium state of the system, making it *relevant*, even below the percolation threshold, thus causing the breaking of SU in the presented simulations.

Performing the simulation in figure 5.2a using a single GTX Titan Black GPU



(a) $q = 8$ (b) $q = 8$



(c) $q = 2$ (d) $q = 2$

Figure 5.2.: Domain growth in a bond-diluted $q = 8$, (a) and (b), and $q = 2$, (c) and (d), Potts models, with lateral sizes $L = 512$ and 2048, respectively. The effective temperatures are $K = 3K_{c,\text{MF}}$ (see text) and broken bond fractions are both $d = 0.2$ and 0.3. The samples consist of $N = 256$ realizations (one MS run). (a),(c): Domain growth against time. The maxima of the R-scale correspond to the respective lateral system size. Inset: Effective growth exponents for $t > 24$ MCS. Error bars are omitted for clarity. (b),(d): Inverse effective exponents against average domain size, as found in [148]. The interval with lower point density in the black plots is caused by omitting 95 % of the available points starting at this time to simplify displaying the plot. Error bars are only displayed for few points.

took more than four months. To effectively investigate the effect of lower temperates, further away from the critical point, simulations of this scale may be necessary. For simulations of systems with other, weaker types of disorder and thus faster growth, shorter runs, around 1×10^8 MCS, may also be sufficient, but larger systems need to be simulated to avoid finite-size effects. In any case to effectively continue this research, enabling the present SkyMC code to make use of multiple GPUs would be helpful.

5.2. Local Scale Invariance in KPZ Surface Growth

Results from this section and section 4.3.1 have been published as an article [145].

In chapter 4 a careful study of the asymptotic properties has been presented. For the roughness scaling, the autocorrelation and the autoresponse, the quality of the available data would allow a very precise calculation of effective exponents. Yet, the estimates of the asymptotic autocorrelation and -response exponents, summarized in section 4.4, carry much larger error margins, due to the form of corrections being uncertain. A next step in KPZ aging studies should thus be to determine the full scaling forms.

It was proposed, that two-point functions, like autocorrelation and autoresponse, for non-equilibrium systems showing scale-invariance can be predicted using an ansatz called local scale-invariance (LSI) [150, 151]. LSI was numerically found to to apply in both pure and disordered two–dimensional Potts models [152, 153] and reaction-diffusion models [72, 73, 154]. Aging properties of diffusive, exactly solvable models with $z = 2$, mean-field like models, which exhibit long-range interactions [155], as well equilibrium interface models like EW [69] and Acetri [156, 157] are also described by LSI.

For the time-integrated autoresponse in the KPZ class, defined in equation (4.24),

$$\chi(t,s) = s^{-a} f_\chi(t/s) \quad , \tag{5.1}$$

LSI predicts the scaling function $f_\chi(t/s)$ to take the form:

$$f_{\chi,\mathrm{LSI}}(t/s) = A_0 (t/s)^{-\lambda_R/z} \left(1 - s/t\right)^{-1-a'} \quad , \tag{5.2}$$

where A_0 is a free parameter and a' is expected to be another universal exponent, like the aging exponent a, for non-equilibrium systems. A different form, adding logarithmic corrections was proposed recently in [151]:

$$\begin{aligned} f_{\chi,\mathrm{L^2LSI}} = (t/s)^{1-\lambda_R/z} \Big[& A_0 \left(1 - (1 - s/t)^{-a'}\right) \\ & + (1 - s/t)^{-a'} \cdot \left(A_1 \ln(1 - s/t) + A_2 \ln^2(1 - s/t)\right)\Big] \quad , \end{aligned} \tag{5.3}$$

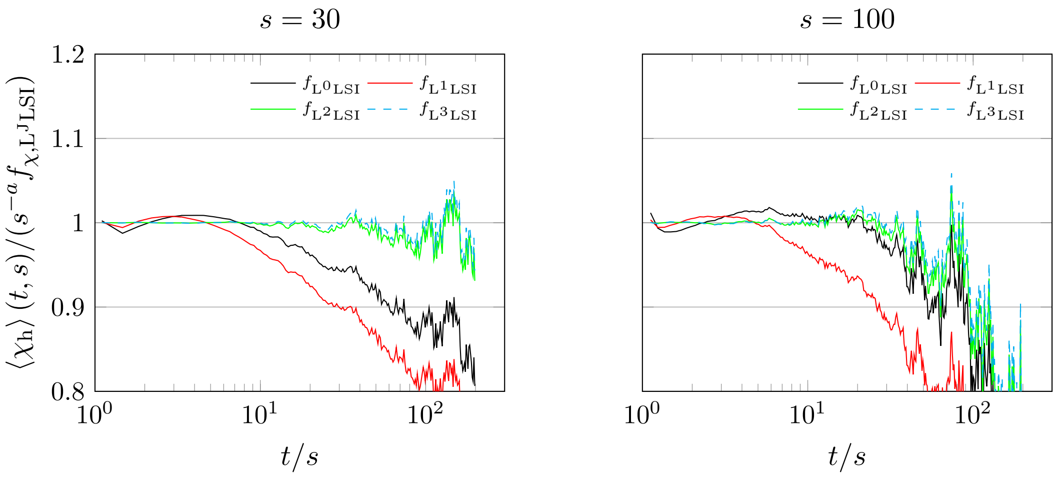


Figure 5.3.: Plots of equation (5.5) for SCA autoresponse calculations with $p = 0.95$ and $q = 0$. Sample sizes are $n_{\mathrm{SCA},s=30} = 23849$ for $s = 30$ (left) and $n_{\mathrm{SCA},s=100} = 12012$ for $s = 100$ (right). Only the interval $1 \leq t/s \leq 10$ was considered for best fits. See figure 4.20 for the response functions $\langle \chi_{\mathrm{heights}} \rangle (t/s)$.

where the sum of logarithmic terms to second order results from the assumption, that the field ϕ is replaced by a duplet and the scaling dimensions of the system must be represented, not as scalars, but as 2×2 matrices. The scaling function (5.3) resembles a form, which contains the first two lowest order correction terms of a logarithmic series to (5.2):

$$f_{\chi,\mathrm{L^JLSI}} = (t/s)^{1-\lambda_R/z} \left[A_0 \left(1 - (1 - s/t)^{-a'}\right) + (1 - s/t)^{-a'} \cdot \sum_{j>0}^{J} A_j \ln^j(1 - s/t) \right] \quad , \tag{5.4}$$

Such a series, when used for fitting, would thus be expected to break off after $J = 2$. However, the assumption of triplets for ϕ, or beyond, would also give physical meaning to some terms with $j \geq 3$. Thus these terms being relevant to describe the data would point to the necessity of higher orders in the extension of LSI. In reference [151] numerical data was presented, suggesting that this theory applies to $1 + 1$-dimensional KPZ.

Only the asymptotic behavior, described by the exponent $\lambda_{R,\mathrm{heights}}/z = 1.98(4)$ was determined in section 4.3.1, but the data available from SCA simulations is actually sufficient to test the L^{2}LSI-theory. Figure 5.3 shows plots of the ratio of data and best fit. This is a visual representation of how well forms for $J \in [0, 3]$

Table 5.1.: Parameters for best fits of $f_{\chi,\mathrm{L^JLSI}}$ forms to KPZ autoresponse functions for $1 \leq t/s \leq 200$. Values for $\lambda_{R,\mathrm{h}}/z$ in parenthesis result from fits considering $q \leq t/s \leq 10$, as presented in figure 5.3. Error margins are not given, because the method employed for fitting does not provide meaningful estimates.

		$\lambda_{R,\mathrm{h}}/z$	a'	A_0	A_1	A_2	A_3
$s = 30$	$f_{\mathrm{L^0LSI}}$ —	1.164 (1.167)	0.016	38.833			
	$f_{\mathrm{L^1LSI}}$ —	1.164 (1.144)	0.023	35.085	0.187		
	$f_{\mathrm{L^2LSI}}$ —	1.224 (1.219)	0.501	4.938	1.772	−0.431	
	$f_{\mathrm{L^3LSI}}$ - - -	1.224 (1.224)	0.505	4.790	1.716	−0.422	−0.004
$s = 100$	$f_{\mathrm{L^0LSI}}$ —	1.186 (1.191)	0.006	102.584			
	$f_{\mathrm{L^1LSI}}$ —	1.165 (1.142)	0.100	14.444	0.844		
	$f_{\mathrm{L^2LSI}}$ —	1.230 (1.224)	0.490	5.544	2.019	−0.472	
	$f_{\mathrm{L^3LSI}}$ - - -	1.230 (1.233)	0.475	5.506	1.914	−0.437	−0.008

describe the data:

$$\frac{\langle \chi_{\mathrm{heights}} \rangle \,(t/s)}{s^{-a} f_{\chi,\mathrm{L^JLSI}}(t/s)} \overset{!}{=} 1 \qquad \text{for } t/s > 1 \quad . \tag{5.5}$$

The non-linear fits for $J > 0$ do not converge using the classical least-squares Levenberg–Marquardt algorithm [158, 159]. To obtain the parameters presented in table 5.1, the Nelder-Mead method [160] was employed, which does not provide statistical error estimates for the fit parameters. Judging by the multitude of local minima a fit can end up in, depending on the initial guesses, and the connected variation in parameter values, the accuracy of the tabulated parameters should be assumed to be no better than 20 %, except for the values of $\lambda_{R,\mathrm{h}}/z$ which vary by less than 5 %.

It is apparent from figure 5.3 that the uncorrected LSI form fails to describe the asymptotic behavior χ_{heights}, giving a $\lambda_{R,\mathrm{h}}/z \approx 1.17$. So does the logarithmic form with $J = 1$. The form with $J = 2$, which is predicted by the theory yields much better fits, with $\lambda_{R,\mathrm{h}}/z \approx 1.22$, which agrees with the asymptotic value obtained earlier. Furthermore, this value is in good agreement with the value determined from tail effective exponents (table 4.5, $\lambda^{\mathrm{tail}}_{R,\mathrm{h}}/z = 1.23(2)$). The parameter fits presented in table 5.1 take into account the observed time interval $1 \leq t/s \leq 200$. When the fit is limited to the interval $1 \leq t/s \leq 10$, the results for $\lambda_{R,\mathrm{h}}$ (values in parenthesis) do not change significantly. This means, that the $f_{\chi,\mathrm{L^2LSI}}$ form describes the corrections, affecting the autoresponse function at early times, well enough to determine the correct asymptotic autoresponse exponent just using early-time data.

The form with $J = 3$ shows marginally better agreement with the data in figure 5.3. In fits to the whole observed time interval, the amplitude A_3 of the added third-order term is severely suppressed (table 5.1). Adding another fit parameter, a slightly better fit is to be expected. The small absolute value of A_3 in relation to A_2 suggests, that the third order does not carry physical meaning, supporting the L^{2}LSI theory.

For both $J = 2$ and 3, the values of the coefficients A_j are similar for all waiting times. This should be the case, since the aging is described by the s^{-a} term in equation (5.1) alone and $f_\chi(t/s)$ should not depend on s explicitly.

For the autocorrelation functions in the KPZ model, no form for $f_{C,\mathrm{L^2LSI}}$ has been proposed yet. One would assume L^{2}LSI to hold for the autocorrelation too, if it holds for the autoresponse, but this remains to be tested.

6. Conclusions and Outlook

This work centered primarily around the efficient and correct execution of lattice Monte-Carlo simulations in general and dynamical properties of surface growth models in the Kardar–Parisi–Zhang (KPZ) universality class in particular. Large-scale simulations require parallel processing and are most efficient, both in terms of energy consumption and computation time, when using massively parallel accelerators like GPUs. In this work an implementation of parallel random-sequential (RS) simulations, which exceeds the performance of actual sequential simulations by more than two orders of magnitude, has been improved to eliminate any measurable correlations with significantly reducing performance. This is an important step, given that most stochastic models of physical systems of assume uncorrelated noise. The widely used approach of sampling lattice sites in a checkerboard pattern (stochastic cellular automaton (SCA)), which can be implemented much more efficiently than RS, was also implemented in this work for GPUs and the effects of its intrinsic correlations have been investigated.

Figure 6.1 illustrates the relations between a part of the results of this work on KPZ surface growth.

(a) It was shown, that when correlations are eliminated finite-time corrections to the roughness growth, which can be observed in the effective growth exponents β_{eff}, are reduced. Smaller corrections reduce uncertainties when extrapolating asymptotic exponents.

(b) The produced high precision data provides evidence for the KPZ ansatz hypothesis and support a new estimate for the universal KPZ growth exponent β with unprecedented precision.

(c) An extension to the restricted solid-on-solid model (RSOS) model provides evidence, that the prediction $\beta < 1/4$ also holds for surfaces which are locally rougher (i. e. slopes between neighboring lattice sites are larger). This result disproved the last claim of the Kim–Kosterlitz hypothesis.

(d) Extensive simulations using SCA dynamics unveiled possible differences in the scaling behavior of with respect to RS dynamics and the update probability, which are, however, too small have been observable in any previous studies found in literature.

(e) The efficient implementation of SCA dynamics allowed to follow the surface growth for large systems ($L = 2^{16}$) over six orders of magnitude. High quality data for the evolution of the distribution of interface heights made finite-size effects observable more than ten times before than the visible onset of the steady state.

(f),(g) The new virtually correlation-free RS simulations allowed determining the auto-correlation properties of the KPZ surface with increased accuracy and confirmed the Kallabis–Krug conjecture, that $\lambda_C = d$, for the investigated case $d = 2$.

(h),(i) Autocorrelation functions have been shown to saturate to a finite asymptotic value under SCA dynamics. For the interface heights, this finite value was found to be approached following the correct power law (PL) form.

(j),(k) The slopes, or lattice-gas variables, have been shown to react more sensitively to correlations, resulting in a larger saturation autocorrelation o. Also, the RS behavior is not recovered after subtracting o, instead a crossover to a slower PL decay was observed.

For comparison, the autocorrelation in the analytically solved Edwards–Wilkinson (EW) class was simulated as well. The known form for the autocorrelation of interface heights was found to be correctly reproduced even under SCA dynamics and no finite asymptotic autocorrelation was observed for heights nor slopes. However, the asymptotic autocorrelation of the latter was again found to be different under SCA dynamics, leading to the conclusion,

(l) that SCA dynamics affects the autocorrelation properties of lattices gases more strongly than that of integrated quantities such as surface heights. It may even affect presumably universal exponents of the lattice gas, while those for the interface remain unaffected.

Finally, the autoresponse of KPZ interfaces has been studied, yielding more precise estimates of the autoresponse and related aging exponents.

(m) The autoresponse exponent λ_R was found to be likely equal to the autocorrelation exponent λ_C. Thus, despite the inequality of the corresponding aging exponents and fact, that the growing surface remains far from equilibrium at all times, there may be an, as of yet unknown, generalized fluctuation-dissipation relation (FDR) connecting correlation and response.

(n) The high precision data obtained from extensive SCA simulations allowed for a successful test of the autoresponse scaling forms predicted by local scale-invariance with logarithmic extensions (L^{2}LSI), thereby numerically supporting this hypothesis.

The computational methods developed here have also been shown to be readily extensible to further studies of Potts models, disordered systems and beyond.

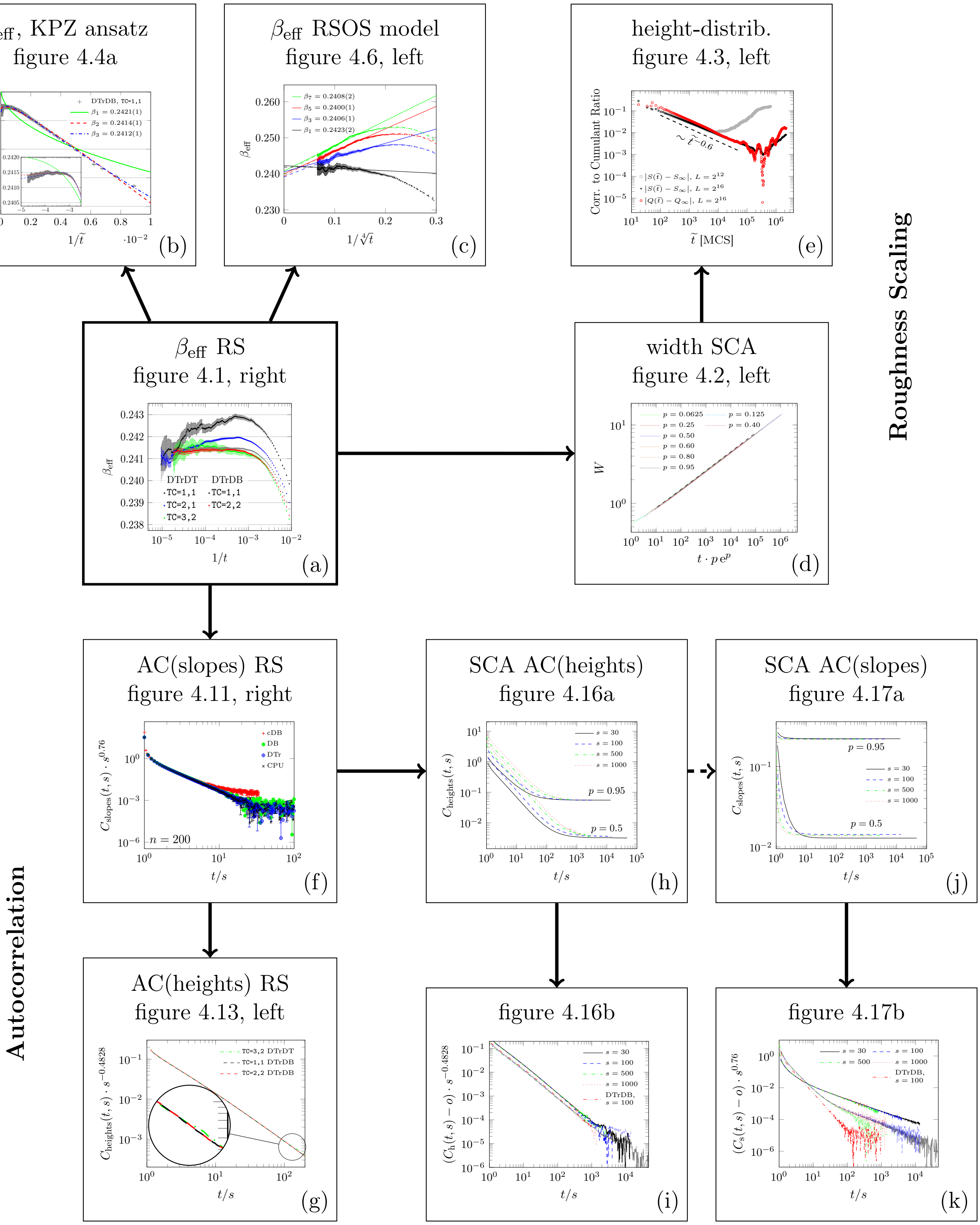


Figure 6.1.: Overview of selected results regarding KPZ surface growth. Unless stated otherwise, all displayed data belong to the octahedron model or the underlying dimer lattice gas. The displayed figures are reproductions out of earlier chapters, please see respective figure captions for details.

Outlook

It was shown, that there are significant differences between RS and SCA simulations, especially for lattice gases. An important technical difference is also present in that the performance of SCA simulations is about 20 times higher. Since SCA cannot replace RS simulations to study certain aspects of surface growth and lattice gas dynamics, this severely limits the size of studies, and thus the numerical precision, which is feasible by comparison. The MS (section 3.4) approach offers a way to increase the performance of RS simulations. In principle, the bit-vectorization enabled by non-local encoding and deterministic site-selection making SCA simulations very efficient could be applied to RS simulations too, by combining it with the MS approach. This would allow RS simulations at almost the performance of SCA runs. The catch is, that, this would require the simultaneous simulation of 1024 systems due to the properties of current GPU architecture. RS simulations are especially important when dynamical properties out-of-equilibrium and far from the steady state are of interest. This requires large system, such as have been used in this work. Assuming a lateral system size of $L = 2^{16}$, 1024 realizations would require 1 TB of memory, which realize any performance gain, must be GPU memory.

Such an endeavour would obviously require multi-GPU support in SkyMC, which is also not very hard to implement. Given about 8 GB of available global memory per GPU, 128 K40 GPU would be required to hold the simulation state. This is an amount of devices actually available at major sites. The payoff form performing simulations of this scale would be, that a single run of about 24 hours would produce a sample almost half the size the largest SCA studies presented in this work, but virtually correlation-free.

Even with fewer computational resources, such a MS would be a powerful tool to perform simulations of smaller systems in the steady state, giving access to a class of problem which have only been addressed parenthetically in this work. This could enable the study of autocorrelation and response functions of KPZ surfaced in the steady state. This is of interest because there even appears to be some disagreement about the exact form of the autocorrelation in the one–dimensional case (compare references [142] and [143]). Since the precise value for the growth exponent β is marginally incompatible with the roughness exponent α calculated in a recent study [133] in which SCA dynamics was used[1], an extensive finite-size scaling study using RS would also be of interest. A new direct estimate agreeing with [133] rather than the value required by the Galilean symmetry would be evidence for a small violation of the latter by KPZ. The observation (e), that finite-size effects can also be observed in the changing shape of the height-distribution orders of magnitude before the system crosses over into the steady state, maybe used to make much large systems accessible to a finite-size scaling study.

Determining the autocorrelation function under SCA dynamics in the steady state for different update probabilities p could help to understand the origin of the finite saturation value o for both interface heights and lattice gas variables. Using the steady state for such a study would allow a more reliable determination of the dependence of o on p, because the waiting time s, as an additional free parameter, is removed from the equation. The dependence of o on s can then be determined separately in the growth regime.

---

[1] This is not completely clear form the paper, but was communicated by G. Parisi directly.

The brief study of the EW case (l) also showed how little is understood about the dynamical properties of lattice gases under the checkerboard updates of SCA, even though SCA dynamics is employed quite frequently in literature. Here, a comparative study of the KPZ and EW cases under both RS and SCA dynamics would be interesting. For one, to learn why the autocorrelation does actually saturate under SCA dynamics in the first case but not in the latter. On the other hand, to check how the different exponents observed under both types of site-selection dynamics are related. A large-scale study of the EW case under RS dynamics, using the MS implementation of the octahedron model motivated initially, could be performed to determine the autocorrelation exponent for the slopes more precisely.

Acknowledgements

I want to thank everyone who supported me during my studies. First and foremost my supervisor Sibylle Gemming, who enabled me to pursue the research presented in this thesis, and my collaborator Géza Ódor who always had many suggestions and open ears for my questions. This research grew out of a side project to my diploma work, which was supervised by Karl-Heinz Heinig, whom I would like to thank too, also for his general suggestions. Furthermore, I would like to thank Malte Henkel, Uwe Täuber, Giorgio Parisi and Herbert Spohn for very helpful discussions about surface growth and universality.

I thank Martin Weigel for inviting me to stay at Coventry University and our continued work together. In this context, I acknowledge partial funding of this research stay by the Erasmus+ program via the Leonardo-Büro Sachsen.

I am grateful for the support by the International Helmholtz Research School for Nanoelectronic Networks (IHRS NanoNet) and my thesis advisory committee: Artur Erbe, Jörg Schuster, Peter Zahn and Sibylle Gemming, whom I hopefully, did not disappoint too much by not writing about molecular transport. I would also like to thank my other colleagues at HZDR and in NANONET for support and useful discussions.

Not least, I would like to especially thank my family and friends for their support in all non-science matters.

A. Coding Details

In this section, logical operations are used, denote by $\wedge$ (and), $\vee$ (or), $\neg$ (not) and $\oplus$ (exclusive or), which shall be defined bit-wise. The operators $\gg$ and $\ll$ denote bit-shifts to the right and left, respectively, where the new bits shifted into the word taking the value 0.

A.1. Bit-Coding

Let b be the number bits required to encode a lattice site and w the word-size in bits. A lattice site s stored in a word W at offset o can then be accessed by

$$s = (W \gg o) \wedge (b - 1) \quad .$$

Writing back arbitrary information requires two first get the corresponding bits in the target word in a defined state (here 0), and then writing the payload:

$$\begin{aligned} W &= W \wedge ([\neg(b-1)] \ll o) \\ W &= W \vee (s \ll o) \end{aligned}$$

However, in models with small numbers of states per site, is is rarely required to write arbitrary data. The most common operation is flipping of bits:

$$W = W \oplus (1 \ll o)$$

A.2. Packing and Unpacking Signed Integers

The smallest integer size available on modern computers is eight-bit. When considering unsigned integers, it is simple to see how sufficiently small values can be stored using a smaller number of bits and be retrieved and used:

$$a \wedge (2^n - 1) = a \qquad 0 < a < 2^n \tag{A.1}$$

This relation does not apply to negative numbers, which, interpreted as unsigned integers, are encoded by very large positive numbers. Still, truncation of this form does conserve all information stored in a signed integer as long as $|a| < 2^n/2$. The following relation allows unpacking of signed integers truncated to length n:

$$a_m = a_n \vee [(0_m - [a_n \gg (n-1)]) \wedge (\neg[2^n - 1])] \quad , \tag{A.2}$$

Where $m > n$ is the number of bits the integer is inflated to. Subscripts denote a number encoded in the respective number of bits.

However, direct encoding of signed integers is not efficient due to the rather complex calculations required for unpacking. It is significantly more efficient to store $a_n = a_m + 2^n/2$, where all valid values $-2^n/2 < a_m < 2^n/2 - 1$ become positive. This severely simplifies the unpack step to be: $a_m = a_n - 2^n/2$. The impact on the overall performance of the GPU implmentation of RSOS (see section 3.4.1) is about 10 % on a GTX Titan Black GPU.

A.3. Random Number Generation

Random numbers for MC simulations are obtained using a PRNG, which is a numerical function generating a sequence of numbers which are random in the sense, that they are not correlated. A PRNG has an internal state, which changes as pseudo-random numbers are generated, and some numerical parameters defining the sequence of number generated. The internal state in seeded at the beginning of a simulation with a numerical value, which is random and with overwhelming probability different for each simulation. The seed used for each simulations is logged, which in principle makes the simulation reproducible.

A simple and often used type of PRNG is the linear congruential generator (LCG), which generates a sequence of numbers according to:

$$x_{i+1} = (a \cdot x_i + c) \mod m \quad , \tag{A.3}$$

where a, c and m are parameters determining the sequence and the generated random number also takes the role of the internal state.

In a parallel MC code, many workers require independent random numbers simultaneously. Using an LCG, it is not possible to chose different parameters for each worker to get different sequences, because the quality of the random numbers very sensitively depends on this choice. It is however possible, to use only one sufficiently long sequence and assign disjunct subsequences to each worker. [161]

One of the best quality PRNGs is the Mersenne Twister [162], which holds an internal state containing about 15 kB of data, providing sequence with a period of $2^{19937} - 1$. A fast implementation of it [163] is used in sequential single-CPU codes, including the host-side of GPU codes, presented in this work. However, the internal state is too large to be efficiently integrated with MC simulations on GPU. However, efficient stand-alone implementations on GPU are available. [164]

The presented parallel MC codes, both for GPU and CPU, use a small version of the Mersenne Twister, called TinyMT [106], which was developed by the same authors as the original Mersenne Twister. Each CPU or SIMT thread maintains its own, randomly seeded, TinyMT state of 128 bits and a polynomial of 96 bits, which defines the generated sequence. The polynomials of all threads are independent of each other.

Bibliography

[1] Metropolis, N. The Beginning of the Mont Carlo Method. *Los Alamos Sci.* 125–130 (1987).

[2] Ulam, S., Richtmyer, R. D. & von Neumann, J. Statistical methods in neutron diffusion. Tech. Rep. LAMS-551, Los Alamos Scientific Laboratory (1947).

[3] Metropolis, N. & Ulam, S. M. The Monte Carlo Method. *J. Am. Stat. Assoc.* **44**, 335–341 (1949).

[4] Newman, M. E. J. & Barkema, G. T. *Monte Carlo Methods in Statistical Physics* (Oxford University Press, 1999), 2002 edn.

[5] Borodin, V., Heinig, K. & Reiss, S. Self-organization kinetics in finite precipitate ensembles during coarsening. *Phys. Rev.B* **56**, 5332 (1997).

[6] Strobel, M. *Modeling and Computer Simulation of Ion Beam Synthesis of Nanostructures*. Ph.D. thesis, TU-Dresden (1999).

[7] Krug, J. Origins of scale invariance in growth processes. *Adv. Phys.* **46**, 139–282 (1997).

[8] Kadanoff, L. P. *et al.* Static Phenomena Near Critical Points: Theory and Experiment. *Rev. Mod. Phys.* **39**, 395–431 (1967).

[9] Binder, K. & Stauffer, D. Theory for the Slowing Down of the Relaxation and Spinodal Decomposition of Binary Mixtures. *Phys. Rev. Lett.* **33**, 1006–1009 (1974).

[10] Hohenberg, P. C. & Halperin, B. I. Theory of dynamic critical phenomena. *Rev. Mod. Phys.* **49**, 435–479 (1977).

[11] Ódor, G. Universality classes in nonequilibrium lattice systems. *Rev. Mod. Phys.* **76**, 663–724 (2004).

[12] Ostwald, W. Über die vermeintliche Isomerie des roten und gelben Quecksilberoxyds und die Oberflächenspannung fester Körper. *Ztschr. Phys. Chem.* **34**, 495–503 (1900).

[13] Lifshitz, I. & Slyozov, V. The kinetics of precipitation from supersaturated solid solutions. *J. Phys. Chem. Solids* **19**, 35–50 (1961).

[14] Plateau, J. A. F. *Statique expérimentale et théorique des liquides soumis aux seules forces moléculaires* (Gauthiers-Villars, 1873).

[15] Rayleigh, J. W. S. On the Instability of Jets. *Proc. London Math. Soc.* **10** (1879).

[16] Boyanovsky, D., de Vega, H. & Schwarz, D. Phase Transitions in the Early and Present Universe. *Annu. Rev. Nucl. Part. Sci.* **56**, 441–500 (2006).

[17] Heinig, K.-H., Müller, T., Schmidt, B., Strobel, M. & Möller, W. Interfaces under ion irradiation: growth and taming of nanostructures. *Appl. Phys. A* **77**, 17–25 (2003).

[18] Müller, T. *Low Energy Ion Beam Synthesis of Si Nanocrystals for Nonvolatile Memories - Modeling and Process Simulations.* Ph.D. thesis, TU-Dresden (2005).

[19] Liedke, B. *Ion beam processing of surfaces and interfaces - Modeling and atomistic simulations.* Ph.D. thesis, TU-Dresden (2011).

[20] Moore, G. E. Cramming more components onto integrated circuits. *Electronics* **38** (1965).

[21] Müller, T. *et al.* Multi-dot floating-gates for nonvolatile semiconductor memories: Their ion beam synthesis and morphology. *Appl. Phys. Lett.* **85**, 2373 (2004).

[22] Heinig, K.-H., Turan, R. *et al.* Rainbow Energy project description (2010). BMBF and TÜBITAK project by HZDR, METU and industry partners.

[23] Tai, M. C. *et al.* Thermal Stability of Nanoporous Raney Gold Catalyst. *Metals* **5**, 1197 (2015).

[24] Ódor, G. *Universality in Nonequilibrium Lattice Systems* (World Scientific, 2008).

[25] Landau, L., Lifšic, E., Sykes, H. & Kearsley, M. *Statistical Physics.* Course of Theoretical Physics (Elsevier Science & Technology, 1980).

[26] Täuber, U. C. *Critical Dynamics* (Cambridge University Press, 2014). Cambridge Books Online.

[27] Kelling, J. & Ódor, G. Extremely large-scale simulation of a Kardar-Parisi-Zhang model using graphics cards. *Phys. Rev. E* **84**, 061150 (2011).

[28] Burgers, J. M. *The nonlinear diffusion equation : asymptotic solutions and statistical problems* (Dordrecht-Holland ; Boston : D. Reidel Pub. Co, 1974). First published in 1973 under title: Statistical problems connected with asymptotic solutions of the one-dimensional nonlinear diffusion equation.

[29] Wagner, C. Theorie der Alterung von Niederschlägen durch Umlösen (Ostwald-Reifung). *Ztschr. Elektrochemie* **65**, 581–591 (1961).

[30] Cahn, J. W. & Hilliard, J. E. Free Energy of a Nonuniform System. I. Interfacial Free Energy. *J. Chem. Phys.* **28**, 258–267 (1958).

[31] Cahn, J. W. On spinodal decomposition. *Acta Metall.* **9**, 795–801 (1961).

[32] Kardar, M., Parisi, G. & Zhang, Y.-C. Dynamic Scaling of Growing Interfaces. *Phys. Rev. Lett.* **56**, 889–892 (1986).

[33] Sun, T., Guo, H. & Grant, M. Dynamics of driven interfaces with a conservation law. *Phys. Rev. A* **40**, 6763–6766 (1989).

[34] Rost, M. & Krug, J. Anisotropic Kuramoto-Sivashinsky Equation for Surface Growth and Erosion. *Phys. Rev. Lett.* **75**, 3894–3897 (1995).

[35] Landau, D. & Binder, K. *A guide to Monte Carlo Simulations in Statistical Physics* (Cambridge, 2005), second edn.

[36] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087–1092 (1953).

[37] Glauber, R. J. Time-Dependent Statistics of the Ising Model. *J. Math. Phys.* **4**, 294–307 (1963).

[38] Ising, E. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik* **31**, 253–258 (1925).

[39] Kawasaki, K. Diffusion Constants near the Critical Point for Time-Dependent Ising Models. I. *Phys. Rev.* **145**, 224–230 (1966).

[40] Röntzsch, L. *Shape Evolution of Nanostructures by Thermal and Ion Beam Processing.* Ph.D. thesis, TU-Dresden (2007).

[41] Potts, R. B. Some generalized order-disorder transformations. *Math. Proc. Camb. Philos. Soc.* **48**, 106–109 (1952).

[42] Wu, F. Y. The Potts model. *Rev. Mod. Phys.* **54**, 235–268 (1982).

[43] Anderson, M., Srolovitz, D., Grest, G. & Sahni, P. Computer simulation of grain growth—I. Kinetics. *Acta Metall.* **32**, 783–791 (1984).

[44] Baxter, R. J. Potts model at the critical temperature. *J. Phys. C Solid State Phys.* **6**, L445 (1973).

[45] Bazavov, A., Berg, B. A. & Dubey, S. Phase transition properties of 3D Potts models. *Nucl. Phys. B* **802**, 421–434 (2008).

[46] Chatelain, C. *Random and Out-of-Equilibrium Potts models.* Habilitation à diriger des recherches, Université de Lorraine (2012). URL `https://tel.archives-ouvertes.fr/tel-00959733`.

[47] Lifshitz, L. Kinetics of Ordering During Second-Order Phase Transitions. *Soviet Physics JETP-USSR* **15**, 939–942 (1962).

[48] Allen, S. M. & Cahn, J. W. A microscopic theory for antiphase boundary motion and its application to antiphase domain coarsening. *Acta Metall.* **27**, 1085–1095 (1979).

[49] Majumder, S. & Das, S. K. Diffusive domain coarsening: Early time dynamics and finite-size effects. *Phys. Rev. E* **84**, 021110 (2011).

[50] Majumder, S. & Das, S. K. Effects of Density Conservation and Hydrodynamics on Aging in Nonequilibrium Processes. *Phys. Rev. Lett.* **111**, 055503 (2013).

[51] Cardy, J. Proceedings of the 20th IUPAP International Conference on Statistical Physics Quenched randomness at first-order transitions. *Phys. A Stat. Mech. Appl.* **263**, 215–221 (1999).

[52] Lippiello, E., Mukherjee, A., Puri, S. & Zannetti, M. Scaling behavior of response functions in the coarsening dynamics of disordered ferromagnets. *EPL* **90**, 46006 (2010).

[53] Fisher, D. S. & Huse, D. A. Nonequilibrium dynamics of spin glasses. *Phys. Rev. B* **38**, 373–385 (1988).

[54] Cugliandolo, L. F. Topics in coarsening phenomena. *Phys. A Stat. Mech. Appl.* **389**, 4360–4373 (2010). Proceedings of the 12th International Summer School on Fundamental Problems in Statistical Physics.

[55] Paul, R., Puri, S. & Rieger, H. Domain growth in random magnets. *EPL* **68**, 881 (2004).

[56] Paul, R., Puri, S. & Rieger, H. Domain growth in Ising systems with quenched disorder. *Phys. Rev. E* **71**, 061109 (2005).

[57] Henkel, M. & Pleimling, M. Superuniversality in phase-ordering disordered ferromagnets. *Phys. Rev. B* **78**, 224419 (2008).

[58] Corberi, F., Lippiello, E., Mukherjee, A., Puri, S. & Zannetti, M. Crossover in growth law and violation of superuniversality in the random-field Ising model. *Phys. Rev. E* **85**, 021141 (2012).

[59] Harris, R. *et al.* Experimental investigation of an eight-qubit unit cell in a superconducting optimization processor. *Phys. Rev. B* **82**, 024511 (2010).

[60] Katzgraber, H. G., Hamze, F. & Andrist, R. S. Glassy Chimeras Could Be Blind to Quantum Speedup: Designing Better Benchmarks for Quantum Annealing Machines. *Phys. Rev. X* **4**, 021008 (2014).

[61] Forster, D., Nelson, D. R. & Stephen, M. J. Large-distance and long-time properties of a randomly stirred fluid. *Phys. Rev. A* **16**, 732–749 (1977).

[62] Zhang, J., Zhang, Y.-C., Alstrøm, P. & Levinsen, M. Modeling forest fire by a paper-burning experiment, a realization of the interface growth mechanism. *Phys. A Stat. Mech. Appl.* **189**, 383–389 (1992).

[63] Landau, D. P. & Family, F. (eds.) *Kinetics of Aggregation and Gelation* (Elsevier, Amsterdam, 1984).

[64] Huergo, M. A. C., Pasquale, M. A., González, P. H., Bolzán, A. E. & Arvia, A. J. Growth dynamics of cancer cell colonies and their comparison with noncancerous cells. *Phys. Rev. E* **85**, 011918 (2012).

[65] Hwa, T. Nonequilibrium dynamics of driven line liquids. *Phys. Rev. Lett.* **69**, 1552–1555 (1992).

[66] Kardar, M. Replica Bethe ansatz studies of two-dimensional interfaces with quenched random impurities. *Nucl. Phys. B* **290**, 582–602 (1987).

[67] Halpin-Healy, T. & Palasantzas, G. Universal correlators and distributions as experimental signatures of $(2 + 1)$-dimensional Kardar-Parisi-Zhang growth. *EPL* **105**, 50001 (2014).

[68] Family, F. & Vicsek, T. Scaling of the active zone in the Eden process on percolation networks and the ballistic deposition model. *J. Phys. A* **18**, L75 (1985).

[69] Edwards, S. F. & Wilkinson, D. R. The Surface Statistics of a Granular Aggregate. *Proc. R. Soc. London, Ser. A* **381**, 17–31 (1982).

[70] Ódor, G., Liedke, B. & Heinig, K.-H. Directed d-mer diffusion describing the Kardar-Parisi-Zhang-type surface growth. *Phys. Rev. E* **81**, 031112 (2010).

[71] Kim, J. M. & Kosterlitz, J. M. Growth in a restricted solid-on-solid model. *Phys. Rev. Lett.* **62**, 2289–2292 (1989).

[72] Henkel, M. Ageing, dynamical scaling and its extensions in many-particle systems without detailed balance. *J. Phys. Condens. Matter* **19**, 065101 (2007).

[73] Ódor, G. Local scale invariance in the parity conserving non-equilibrium kinetic Ising model. *J. Stat. Mech.* **2006**, L11002 (2006).

[74] Ódor, G., Liedke, B. & Heinig, K.-H. Mapping of $2+1$ dimensional KPZ growth onto driven lattice gas model of dimers. *Phys. Rev. E* **79**, 021125 (2009).

[75] Plischke, M., Rácz, Z. & Liu, D. Time-reversal invariance and universality of two-dimensional growth models. *Phys. Rev. B* **35**, 3485–3495 (1987).

[76] Meakin, P., Ramanlal, P., Sander, L. M. & Ball, R. C. Ballistic deposition on surfaces. *Phys. Rev. A* **34**, 5091–5103 (1986).

[77] Ódor, G., Liedke, B. & Heinig, K.-H. Surface pattern formation and scaling described by conserved lattice gases. *Phys. Rev. E* **81**, 051114 (2010).

[78] Alves, S. G. & Ferreira, S. C. Scaling, cumulant ratios, and height distribution of ballistic deposition in $3+1$ and $4+1$ dimensions. *Phys. Rev. E* **93**, 052131 (2016).

[79] Pagnani, A. & Parisi, G. Multisurface coding simulations of the restricted solid-on-solid model in four dimensions. *Phys. Rev. E* **87**, 010102 (2013).

[80] Hosseinabadi, S., Movahed, S. M. S., Rajabpour, M. A. & Allaei, S. M. V. Dynamical and geometrical exponents of self-affine rough surfaces on regular and random lattices. *J. Stat. Mech.* **2014**, P12023 (2014).

[81] Kim, J. M., Kosterlitz, J. M. & Ala-Nissila, T. Surface growth and crossover behaviour in a restricted solid-on-solid model. *J. Phys. A* **24**, 5569 (1991).

[82] Alves, S. G., Oliveira, T. J. & Ferreira, S. C. Universality of fluctuations in the Kardar-Parisi-Zhang class in high dimensions and its upper critical dimension. *Phys. Rev. E* **90**, 020103 (2014).

[83] Kim, J. M. Restricted solid-on-solid model in d = 2 + 1 dimension with various restriction parameters N. *J. Korean Phys. Soc.* **67**, 1529–1532 (2015).

[84] Swendsen, R. H. & Wang, J.-S. Nonuniversal critical dynamics in Monte Carlo simulations. *Phys. Rev. Lett.* **58**, 86–88 (1987).

[85] Wolff, U. Collective Monte Carlo updating for spin systems. *Phys. Rev. Lett.* **62**, 361 (1989).

[86] Ren, RC and Orkoulas, G. Acceleration of Markov chain Monte Carlo simulations through sequential updating. *J. Chem. Phys.* **124** (2006).

[87] Wolfram, S. *A new kind of science*, vol. 1 (Wolfram Media Champaign, IL, 2002).

[88] Preis, T. *et al.* GPU accelerated Monte Carlo simulation of the 2D and 3D Ising model. *J. Comp. Phys.* **228**, 4468–4477 (2009).

[89] Weigel, M. Performance potential for simulating spin models on GPU. *J. Comp. Phys.* **231**, 3064–3082 (2012).

[90] Block, B., Virnau, P. & Preis, T. Multi-GPU accelerated multi-spin Monte Carlo simulations of the 2D Ising model. *Comp. Phys. Comm.* **181**, 1549–1556 (2010).

[91] Kelling, J., Ódor, G., Nagy, M. F., Schulz, H. & Heinig, K. Comparison of different parallel implementations of the 2+1-dimensional KPZ model and the 3-dimensional KMC model. *Eur. Phys. J.: Spec. Top.* **210**, 175–187 (2012). 10.1140/epjst/e2012-01645-8.

[92] Rajewsky, N., Santen, L., Schadschneider, A. & Schreckenberg, M. The Asymmetric Exclusion Process: Comparison of Update Procedures. *J. Stat. Phys.* **92**, 151–194 (1998).

[93] Kelling, J. *Kinetic Monte Carlo Simulations on Self-organization of Nanostructures Accelerated by Massive Parallelization.* Master's thesis, TU-Dresden (2012).

[94] Bortz, A. B., Kalos, M. H. & Lebowitz, J. L. A new algorithm for Monte Carlo simulations of Ising spin systems. *J. Comp. Phys.* **17**, 10–18 (1975).

[95] Shim, Y. & Amar, J. Semirigorous synchronous sublattice algorithm for parallel kinetic Monte Carlo simulations of thin film growth. *Phys. Rev. B* **71**, 125432 (2005).

[96] Shim, Y. & Amar, J. Rigorous synchronous relaxation algorithm for parallel kinetic Monte Carlo simulations of thin film growth. *Phys. Rev. B* **71**, 115436 (2005).

[97] Plimpton, S. *et al.* Crossing the Mesoscale No-Man's Land via Parallel Kinetic Monte Carlo. Tech. Rep., Sandia National Laboratories (2009). URL `http://www.sandia.gov/~sjplimp/kmc.html`.

[98] Dall, J. & Sibani, P. Faster Monte Carlo simulations at low temperatures. The waiting time method. *Comput. Phys. Commun.* **141**, 260–267 (2001).

[99] Amdahl, G. M. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In *Proceedings of the April 18-20, 1967, Spring Joint Computer Conference*, AFIPS '67 (Spring), 483–485 (ACM, New York, NY, USA, 1967). URL `http://doi.acm.org/10.1145/1465482.1465560`.

[100] Lulli, M., Bernaschi, M. & Parisi, G. Highly optimized simulations on single- and multi-GPU systems of the 3D Ising spin glass model. *Comput. Phys. Commun.* **196**, 290–303 (2015).

[101] Creutz, M., Jacobs, L. & Rebbi, C. Monte Carlo study of Abelian lattice gauge theories. *Phys. Rev. D* **20**, 1915–1922 (1979).

[102] Ito, N. & Kanada, Y. An effective algorithm for the Monte Carlo simulation of the Ising model on a vector processor. *Supercomputer* **3** (1988).

[103] Ódor, G., Kelling, J. & Gemming, S. Aging of the (2+1)-dimensional Kardar-Parisi-Zhang model. *Phys. Rev. E* **89**, 032146 (2014).

[104] Kelling, J., Heinig, K.-H. & Gemming, S. GPU-based Atomistic Simulations on spatio-temporal experimental Scales. GPU Technology Conference (NVIDIA, 2014). URL `http://on-demand.gputechconf.com/gtc/2014/poster/pdf/P4154_kinetic_monte-carlo_phenomena_cellular.pdf`.

[105] Kelling, J., Ódor, G. & Gemming, S. Bit-Vectorized GPU Implementation of a Stochastic Cellular Automaton Model for Surface Growth. In *2016 IEEE International Conference on Intelligent Engineering Systems, 2016. INES '16* (IEEE, 2016). URL `https://doi.org/10.1109/INES.2016.7555127`.

[106] TinyMT. URL `http://www.math.sci.hiroshima-u.ac.jp/~%20m-mat/MT/TINYMT/index.html`.

[107] GCC Vector extensions Documtation. URL `https://gcc.gnu.org/onlinedocs/gcc/Vector-Extensions.html`.

[108] Ferrenberg, A., Landau, D. & Binder, K. Statistical and systematic errors in Monte Carlo sampling. *J. Stat. Phys.* **63**, 867–882 (1991).

[109] Marinari, E., Pagnani, A. & Parisi, G. Critical exponents of the KPZ equation via multi-surface coding numerical simulations. *J. Phys. A* **33**, 8181 (2000).

[110] Rieger, H. Fast vectorized algorithm for the Monte Carlo simulation of the random field Ising model. *J. Stat. Phys.* **70**, 1063–1073 (1993).

[111] Hukushima, K. & Nemoto, K. Exchange Monte Carlo Method and Application to Spin Glass Simulations. *J. Phys. Soc. Jpn.* **65**, 1604–1608 (1996).

[112] Stroustrup, B. *The C++ Programming Language* (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000), 3rd edn.

[113] Kawasaki, K. Diffusion Constants near the Critical Point for Time-Dependent Ising Models. III. Self-Diffusion Constant. *Phys. Rev.* **150**, 285–290 (1966).

[114] Elçi, E. M. (2016). Private communication.

[115] Kelling, J., Odor, G. & Gemming, S. Dynamical universality classes of simple growth and lattice gas models. *ArXiv e-prints* (2017). `1701.03638`.

[116] Kelling, J., Ódor, G. & Gemming, S. Suppressing correlations in massively parallel simulations of lattice models. *Computer Physics Communications* **220**, 205–211 (2017).

[117] Deker, U. & Haake, F. Fluctuation-dissipation theorems for classical processes. *Phys. Rev. A* **11**, 2043–2056 (1975).

[118] Henkel, M., Noh, J. D. & Pleimling, M. Phenomenology of aging in the Kardar-Parisi-Zhang equation. *Phys. Rev. E* **85**, 030102 (2012).

[119] Forrest, B. M. & Tang, L.-H. Surface roughening in a hypercube-stacking model. *Phys. Rev. Lett.* **64**, 1405–1408 (1990).

[120] Schwartz, M. & Perlsman, E. Upper critical dimension of the Kardar-Parisi-Zhang equation. *Phys. Rev. E* **85**, 050103 (2012).

[121] Marinari, E., Pagnani, A., Parisi, G. & Rácz, Z. Width distributions and the upper critical dimension of Kardar-Parisi-Zhang interfaces. *Phys. Rev. E* **65**, 026136 (2002).

[122] Foltin, G., Oerding, K., Rácz, Z., Workman, R. L. & Zia, R. K. P. Width distribution for random-walk interfaces. *Phys. Rev. E* **50**, R639–R642 (1994).

[123] Halpin-Healy, T. (2+1)-Dimensional Directed Polymer in a Random Medium: Scaling Phenomena and Universal Distributions. *Phys. Rev. Lett.* **109**, 170602 (2012).

[124] Oliveira, T. J., Alves, S. G. & Ferreira, S. C. Kardar-Parisi-Zhang universality class in $(2 + 1)$ dimensions: Universal geometry-dependent distributions and finite-time corrections. *Phys. Rev. E* **87**, 040102 (2013).

[125] Paiva, T. & Aarão Reis, F. Height and roughness distributions in thin films with Kardar–Parisi–Zhang scaling. *Surf. Sci.* **601**, 419–424 (2007).

[126] Aarão Reis, F. D. A. Universality in two-dimensional Kardar-Parisi-Zhang growth. *Phys. Rev. E* **69**, 021610 (2004).

[127] Ferrari, P. L. & Frings, R. Finite Time Corrections in KPZ Growth Models. *J. Stat. Phys.* **144**, 1123 (2011).

[128] Sasamoto, T. & Spohn, H. One-Dimensional Kardar-Parisi-Zhang Equation: An Exact Solution and its Universality. *Phys. Rev. Lett.* **104**, 230602 (2010).

[129] Alves, S. G., Oliveira, T. J. & Ferreira, S. C. Non-universal parameters, corrections and universality in Kardar–Parisi–Zhang growth. *J. Stat. Mech.* **2013**, P05007 (2013).

[130] Kelling, J., Ódor, G. & Gemming, S. Universality of (2+1)-dimensional restricted solid-on-solid models. *Phys. Rev. E* **94**, 022107 (2016).

[131] Oliveira, T. J., Alves, S. G. & Ferreira, S. C. Kardar-Parisi-Zhang universality class in $(2 + 1)$ dimensions: Universal geometry-dependent distributions and finite-time corrections. *Phys. Rev. E* **87**, 040102 (2013).

[132] Alves, S. G., Oliveira, T. J. & Ferreira, S. C. Origins of scaling corrections in ballistic growth models. *Phys. Rev. E* **90**, 052405 (2014).

[133] Pagnani, A. & Parisi, G. Numerical estimate of the Kardar-Parisi-Zhang universality class in (2+1) dimensions. *Phys. Rev. E* **92**, 010101 (2015).

[134] Aarão Reis, F. D. A. Numerical study of roughness distributions in nonlinear models of interface growth. *Phys. Rev. E* **72**, 032601 (2005).

[135] Rodrigues, E. A., Mello, B. A. & Oliveira, F. A. Growth exponents of the etching model in high dimensions. *J. Phys. A* **48**, 035001 (2015).

[136] Wio, H. S., Revelli, J. A., Deza, R. R., Escudero, C. & de la Lama, M. S. KPZ equation: Galilean-invariance violation, consistency, and fluctuation-dissipation issues in real-space discretization. *EPL* **89**, 40008 (2010).

[137] Krech, M. Short-time scaling behavior of growing interfaces. *Phys. Rev. E* **55**, 668–679 (1997).

[138] Krech, M. Erratum: Short-time scaling behavior of growing interfaces [Phys. Rev. E **55** , 668 (1997)]. *Phys. Rev. E* **56**, 1285–1285 (1997).

[139] Kallabis, H. & Krug, J. Persistence of Kardar-Parisi-Zhang interfaces. *EPL* **45**, 20 (1999).

[140] Daquila, G. L. & Täuber, U. C. Slow relaxation and aging kinetics for the driven lattice gas. *Phys. Rev. E* **83**, 051107 (2011).

[141] Schwartz, M. & Edwards, S. Stretched exponential in non-linear stochastic field theories . *Phys. A Stat. Mech. Appl.* **312**, 363–368 (2002).

[142] Prähofer, M. & Spohn, H. Exact Scaling Functions for One-Dimensional Stationary KPZ Growth. *J. Stat. Phys.* **115**, 255–279 (2004). `cond-mat/0212519`.

[143] Katzav, E. & Schwartz, M. Numerical evidence for stretched exponential relaxations in the Kardar-Parisi-Zhang equation. *Phys. Rev. E* **69**, 052603 (2004).

[144] Röthlein, A., Baumann, F. & Pleimling, M. Symmetry-based determination of space-time functions in nonequilibrium growth processes. *Phys. Rev. E* **74**, 061604 (2006).

[145] Kelling, J., Odor, G. & Gemming, S. Local scale-invariance of the 2+1 dimensional Kardar–Parisi–Zhang model. *J. Phys. A* **50**, 12LT01 (2017).

[146] Das, S. K. & Puri, S. Dynamics of phase separation in multicomponent mixtures. *Phys. Rev. E* **65**, 026141 (2002).

[147] Blanchard, T. & Picco, M. Frozen into stripes: Fate of the critical Ising model after a quench. *Phys. Rev. E* **88**, 032131 (2013).

[148] Corberi, F., Lippiello, E., Mukherjee, A., Puri, S. & Zannetti, M. Growth law and superuniversality in the coarsening of disordered ferromagnets. *J. Stat. Mech.* **2011**, P03016 (2011).

[149] Corberi, F., Lippiello, E., Mukherjee, A., Puri, S. & Zannetti, M. Scaling in the aging dynamics of the site-diluted Ising model. *Phys. Rev. E* **88**, 042129 (2013).

[150] Henkel, M., Pleimling, M., Godrèche, C. & Luck, J.-M. Aging, Phase Ordering, and Conformal Invariance. *Phys. Rev. Lett.* **87**, 265701 (2001).

[151] Henkel, M. On logarithmic extensions of local scale-invariance. *Nucl. Phys. B* **869**, 282–302 (2013).

[152] Lorenz, E. & Janke, W. Numerical tests of local scale invariance in ageing q-state Potts models. *EPL* **77**, 10003 (2007).

[153] Henkel, M. & Pleimling, M. Ageing in disordered magnets and local scale invariance. *EPL* **76**, 561 (2006).

[154] Enss, T., Henkel, M., Picone, A. & Schollwöck, U. Ageing phenomena without detailed balance: the contact process. *J. Phys. A* **37**, 10479 (2004).

[155] Henkel, M. & Pleimling, M. *Non-Equilibrium Phase Transitions: Volume 2: Ageing and Dynamical Scaling Far from Equilibrium*. Theoretical and Mathematical Physics (Springer Netherlands, 2010).

[156] Henkel, M. & Durang, X. Spherical model of growing interfaces. *J. Stat. Mech.* **2015**, P05022 (2015).

[157] Henkel, M. Dynamical Symmetries and Causality in Non-Equilibrium Phase Transitions. *Symmetry* **7**, 2108 (2015).

[158] Levenberg, K. A method for the solution of certain non-linear problems in least squares. *Q. J. Appl. Math.* **II**, 164–168 (1944).

[159] Marquardt, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *J. Soc. Ind. Appl. Math.* **11**, 431–441 (1963).

[160] Nelder, J. A. & Mead, R. A Simplex Method for Function Minimization. *Comput. J.* **7**, 308–313 (1965).

[161] Manssen, M., Weigel, M. & Hartmann, A. Random number generators for massively parallel simulations on GPU. *Eur. Phys. J.: Spec. Top.* **210**, 53–71 (2012). 10.1140/epjst/e2012-01637-8.

[162] Matsumoto, M. & Nishimura, T. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *TOMACS* **8**, 3–30 (1998).

[163] Saito, M. & Matsumoto, M. *A PRNG Specialized in Double Precision Floating Point Numbers Using an Affine Transition*, 589–602 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2009). URL `http://dx.doi.org/10.1007/978-3-642-04107-5_38`.

[164] Barash, L. & Shchur, L. PRAND: {GPU} accelerated parallel random number generation library: Using most reliable algorithms and applying parallelism of modern {GPUs} and {CPUs}. *Comput. Phys. Commun.* **185**, 1343–1353 (2014).

[165] NVIDIA. *NVIDIA CUDA Programming Guide*, 8.0 edn. (2016). URL `http://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html`.

List of Figures

2.1. Molecular dynamics vs lattice Monte-Carlo 18
2.2. Illustration of a two–dimensional surface 22
2.3. Schematics for roof-top and octahedron model. 24

3.1. DD schematics 36
3.2. RS octahedron model benchmarks 40
3.3. Comparison of local and non-local encoding of the octahedron model. 42
3.4. Non-local SCA implementation of the octahedron model. 43
3.5. SCA octahedron model benchmarks 45
3.6. SkyMC Benchmarks 53

4.1. KPZ state roughness growth under RS dynamics 61
4.2. KPZ state roughness growth under SCA dynamics 62
4.3. KPZ growth phase height distribution 64
4.4. KPZ ansatz: RS vs. SCA 66
4.5. KPZ steady state width distribution 69
4.6. Roughness (W^2) of an RSOS surface in the scaling regieme. 70
4.7. RSOS steady-state finite-size scaling. 72
4.8. RSOS scaling/steady-state collapse 74
4.9. RSOS height distribution in the steady-state 75
4.10. RSOS steady-state finite-size scaling, DD self-consitency. 76
4.11. KPZ autocorreclations with different device-layer DDs for RS. 79
4.12. Overview of RS Autocorrelation results using DTrDT. 81
4.13. Overview of RS Autocorrelation results using DTrDB. 83
4.14. Autocorreclation results for heights from RS calculations. 85
4.15. Autocorreclation results for slopes from RS calculations. 86
4.16. Autocorreclation results for heights of SCA calculations. 88
4.17. Autocorreclation results for slopes of SCA calculations. 90
4.18. EW autocorrelation of heights 92
4.19. EW autocorrelation of slopes 93
4.20. KPZ autoresponse of heights 95
4.21. KPZ autoresponse of slopes 98

5.1. Domain growth in pure Potts models: CPU vs. GPU 102
5.2. Domain growth in bond-diluted Potts models 104
5.3. KPZ autoresponse of heights vs. L^{J}LSI 106

6.1. Overview of KPZ results 111

List of Tables

3.1. Compute devices used for benchmarks. 30

4.1. KPZ ansatz: fit parameters for the octahedron model 68
4.2. Estimates for scaling exponent β for RSOS. 71
4.3. Summary of estimates for KPZ universal exponents 77
4.4. SCA autocorrelation limits o 87
4.5. Estimates for the autoresponse of heights exponent. 97
4.6. Sample standard deviation of autoresponse of heights data. 99
4.7. Summary of KPZ autocorrelation and -response exponents. 99
4.8. Summary of KPZ aging exponents. 100

5.1. Fit parameters for L^{J}LSI of KPZ autoresponse of heights 107

Glossary

CUDA NVIDIA's proprietary low-level Application Programming Interface (API) and C/C++-dialect for GPGPU programming [165]. 28

MPI Message Passing Interface. An API for interprocess communication, both for shared memory systems and clusters. 33

SIMD Execution paradigm in vector processing: A vector instruction performs the same operation on all elements (scalar integer or floating point number) of a vector (SIMD-word). SIMD instruction sets on x86 CPUs are (various versions of) MMX, 3d-Now!, SSE, AVX. 31

SIMT SIMD-like paradigm used on GPUs, where instead of exposing vector instructions to the programmer or relying on automatic vectorization by the compiler, each element of the elements of SIMD vectors are presented as threads which execute instructions in lock-step. 31

warp SIMT unit of 32 threads on NVIDIA GPUs. The equivalent on AMD GPUs is called *wavefront*, containing 64 threads. 31

Acronyms

AC autocorrelation. 81
API Application Programming Interface. 135
ASEP asymmetric exclusion process. 24

BD ballistic deposition model. 71
BKL Bortz–Kalos–Lebowitz. 28

cDB coarse dead border. 36, 78
CPU central processing unit. 6, 10
CUDA Compute unified device architecture. 28, *Glossary:* CUDA

DB dead border. 5, 35, 36
DD domain decomposition. 28, 35
DT double tiling. 36, 37, 80
DTr DT DD with random origin. 5, 36, 37, 80
DTrDB DTr at device level and single-hit DB at block level. 40
DTrDT DTr at device level and single-hit DT at block level. 40
DTrDTr DTr at device level and single-hit DTr at block level. 60, 61

EW Edwards–Wilkinson. 6, 11, 110

FBC fixed boundary conditions. 35
FDR fluctuation-dissipation relation. 57, 110
FOM figure of merit. 97

GOE Gaussian orthogonal ensemble. 65
GPU graphics processing unit. 2, 10

KK Kim–Kosterlitz. 22, 25

KLMC 3D kinetic Metropolis lattice Monte Carlo. 19, 27

KPZ Kardar–Parisi–Zhang. 2, 11, 16, 21, 109

KS Kuramoto–Shivashinsky. 16

LCG linear congruential generator. 44, 118

LSB least significant bit. 43

LSI local scale-invariance. 105

MBE molecular beam epitaxy. 16

MC Monte Carlo. 9, 13

MCS Monte Carlo step. 35

MD molecular dynamics. 10

MPI Message Passing Interface. 33, *Glossary:* MPI

MS multi-surface coding. 47

NN nearest neighbor. 18

PBC periodic boundary conditions. 36

PL power law. 14, 110

RNG random number generator. 31

RS random-sequential. 6, 11, 27, 34, 109

RSOS restricted solid-on-solid model. 2, 11, 58, 109

S/N signal-to-noise ratio. 71

SCA stochastic cellular automaton. 2, 5, 6, 12, 27, 109

SIMD single instruction multiple data. 31, *Glossary:* SIMD

SIMT single instruction multiple thread. 31, *Glossary:* SIMT

SU super-universality. 20, 102

TASEP totally asymmetric exclusion process. 24

TC thread cell. 39

TDP thermal design power. 30

WTM waiting time method. 28

Selbstständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe. Die aus fremden Quellen direkt oder indirekt übernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Diese Arbeit wurde am Helmholtz–Zentrum Dresden–Rossendorf e. V. unter wissenschaftlicher Betreuung von Prof. Dr. Sibylle Gemming.

Dresden, 13. August 2018

Jeffrey Kelling

Curriculum Vitæ

Jeffrey Kelling
born 7th February 1987 in Rüdersdorf, Germany
Helmholtz-Zentrum Dresden–Rossendorf, Department of Information Services and Computing, Computational Science

Work Experience (2009 – present)

03/2016 – present	Scientist at Helmholtz-Zentrum Dresden–Rossendorf (HZDR), Department of Computational Science
05/2012 – 02/2016	Scientist at HZDR, Institute for Ion beam physics, Non-Equilibrium Thermodynamics
06/2012 – 04/2013	Scientist at TU-Chemnitz, Institute of Physics, Theorie ungeordneter Systeme
10/2009 – 02/2012	Tutoring, TU-Dresden, *Programming for Physicists*
09/2009 – 04/2011	student assistant at HZDR

Education (2000 – 2012)

2012 – 2018	PhD studies, physics, TU-Chemnitz, International Helmholtz Research School for Nanoelectronic Networks (IHRS NanoNet)
2011 – 2012	Diplomathesis at Helmholtz-Zentrum Dresden–Rossendorf / TU-Dresden: *Kinetic Monte Carlo Simulations on Self-organization of Nanostructures Accelerated by Massive Parallelization*
2006 – 2012	study of physics (Diplom), TU-Dresden
2000 – 2006	Carl Bechstein Gymnasium Erkner, Abitur

Schools, Stays Abroad

11/2015 – 01/2016	Applied Mathematics Research Centre, Coventry University, Coventry, United Kingdom
03/2014	IFF spring school: *Computing Solids: Models, Ab-initio Methods and Supercomputing*, Forschungszentrum Jülich, Jülich, Germany
08/2013	Institute of Technical Physics and Materials Science, Centre for Energy Research, Budapest, Hungary
10/2012	cecam school: *Response treatment for the dynamical properties of materials with the ABINIT package*, ETH Zürich, Zurich, Switzerland

List of Publications (Kelling, J.)

Journal and Proceedings Articles

2018	Kelling, J., Ódor, G., Gemming, S.: *Dynamical universality classes of simple growth and lattice gas models*, Journal of Physics A **51**, 035003, (2018) `https://arxiv.org/abs/1701.03638`
2017	Kelling, J., Ódor, G., Gemming, S.: *Suppressing correlations in massively parallel simulations of lattice models*, Computer Physics Communications **220**, 205-211 (2017) `http://arxiv.org/abs/1705.01022`
2017	Kelling, J., Ódor, G., Gemming, S.: *Local scale-invariance of the 2+1 dimensional Kardar–Parisi–Zhang model*, Journal of Physics A **50**, 12LT01 (2017) `https://arxiv.org/abs/1609.05795`
2017	Kelling, J., Zahn P., Schuster J., Gemming, S.: *Elastic and piezoresistive properties of nickel carbides from first principles*, Physical Review B **95**, 024113 (2017) `https://arxiv.org/abs/1604.00328`
2016	Kelling, J., Ódor, G., Gemming, S.: *Bit-Vectorized GPU Implementation of a Stochastic Cellular Automaton Model for Surface Growth*, IEEE International Conference on Intelligent Engineering Systems (2016) `https://arxiv.org/abs/1606.00310`
2016	Kelling, J., Ódor, G., Gemming, S.: *Universality of (2+1)-dimensional restricted solid-on-solid models*, Physical Review E **94**, 022107 (2016) `https://arxiv.org/abs/1605.02620`
2014	Ódor, G., Kelling, J., Gemming, S.: *Aging of the (2+1)-dimensional Kardar-Parisi-Zhang model*, Physical Review E **89**, 032146 (2014) `https://arxiv.org/abs/1312.6029`
2012	Kelling, J., Ódor, G., Ferenc Nagy, M., Schulz, H., Heinig, K.-H.: *Comparison of different parallel implementations of the 2+1-dimensional KPZ model and the 3-dimensional KMC model*, The European Physical Journal - Special Topics **210**, 175-187 (2012) `https://arxiv.org/abs/1204.5072`
2012	Ódor, G., Liedke, B., Heinig, K.-H., Kelling, J.: *Ripples and dots generated by lattice gases*, Applied Surface Science **258(9)**, 4186-4190 (2012)
2011	Kelling, J., Ódor, G.: *Extremely large-scale simulation of a Kardar–Parisi–Zhang model using graphics cards*, Physical Review E **84**, 061150 (2011) `https://arxiv.org/abs/1110.6745`

Invited Talks

10/2017	Kelling, J., Heinig, K.-H., Weigel, M., Gemming, S. *GPU-Accelerated Kinetic Lattice Monte Carlo for Experimental-Scale Studies* TYC@Imperial, London, England
09/2016	Kelling, J., Ódor, G., Heinig, K. H., Weigel, M., Gemming, S. *Pushing the Limits of Lattice Monte-Carlo Simulations using GPUs* Perspectives of GPU computing in Science, Rome, Italy
05/2015	Kelling, J. *C++11/14 features relevant in GPGPU APIs* GPU-Day 2015—The Future of Many-Core Computing in Science, Budapest, Hungary:
05/2015	Kelling, J., Ódor, G., Heinig, K.-H., Gemming, S. *Efficient Large Scale Simulation of Stochastic Lattice Models on GPUs* GPU Day 2015—The Future of Many-Core Computing in Science, Budapest, Hungary

Talks

10/2017	Kelling, J., Heinig, K.-H., Weigel, M., Gemming, S. *GPU-Accelerated Kinetic Lattice Monte Carlo for Experimental-Scale Studies* MRS Fall Meeting, Boston, USA
05/2017	Kelling, J., Weigel, M., Ódor, G., Gemming, S. *Efficient Correlation-Free Many-States Lattice Monte Carlo on GPUs* GTC, San Jose, CA, USA
09/2016	Kelling, J., Heinig, K.-H., Gemming, S. *Experimental-Scale Kinetic Lattice Monte-Carlo Studies on GPU* E-MRS Fall Meeting, Warsaw, Poland
07/2016	Kelling, J., Ódor, G., Gemming, S. *Aging In The* $(2+1)$*–Dimensional Kardar–Parisi–Zhang Model Under Various Dimer Lattice-Gas Dynamics* Stat'Phys 26—Statistical Physics Conference Satellite Non-equilibrium dynamics in classical and quantum systems: From quenches to slow relaxations, Pont-à-Mousson, France
09/2015	Kelling, J., Sendler, T., Erbe, A., Gemming, S. *Electronic Transport through Au-contacted, Thiol-terminated, PEEB* NanoNet Workshop, Rathen, Germany
03/2015	Kelling, J., Heinig, K.-H., Gemming, S. *Investigating Spinodal Decomposition and Coarsening using Massively Parallel Kinetic Metropolis Lattice Monte-Carlo Simulations* MECO40, Esztergom, Hungary
05/2013	Kelling, J., Heinig, K.-H. *Performing kinetic lattice Monte-Carlo simulations of far-from-equilibrium processes on GPUs* 2nd International Symposium "Computer Simulations on GPU", Freudenstadt, Germany

Posters

04/2016	Kelling, J., Ódor, G., Gemming, S. *Non-Local Lattice Encoding for Bit-Vectorized Cellular Automata GPU Implementations* GTC, San Jose, CA, USA
03/2016	Kelling, J., Ódor, G., Gemming, S. *Aging Universality Classes in Surface Growth Models* DPG Sping Meeting, Sektion Kondensierte Materie, Dresden, Germany

02/2016	Kelling, J., Ódor, G., Gemming, S. *Aging in the* $(2 + 1)$*–Dimensional Kardar–Parisi–Zhang Model under Various Dimer Lattice-Gas Dynamics* MECO41, Vienna, Austria
09/2015	Kelling, J., Sendler, T., Erbe, A., Gemming, S. *Electronic Transport through Au-contacted PEEB* NanoNet Workshop, Rathen, Germany
03/2015	Schulz, H., Kelling, J., Ódor, G., Ódor, G., Ferenc, Nagy, M. *Simulation of Surface Growth and Lattices Gases Using GPUs* GTC, San Jose, CA, USA and MECO40, Esztergom, Hungary
03/2015	Kelling, J., Ódor, G., Gemming, S. *Handling Domain Decomposition in Massively Parallel Implementations of Stochastic Lattice Models* GTC, San Jose, CA, USA and MECO40, Esztergom, Hungary
03/2015	Kelling, J., Kerbusch, J., Erbe, A., Dietsche, R., Ganteför, G., Scheer, E., Zahn, P., Gemming, S. *Transport Calculations for* Si_4 *Clusters with Gold single Atom Contacts* DPG Sping Meeting, Sektion Kondensierte Materie, Berlin, Germany
09/2014	Kelling, J., Gemming, S. *Electronic Transport through Au-contacted PEEB* NanoNet Workshop, Rathen, Germany
04/2014	Kelling, J., Zahn, P., Gemming, S. *Elastic Properties of Nickel Carbides* DPG Sping Meeting, Sektion Kondensierte Materie, Dresden, Germany
04/2014	Kelling, J., Heinig, K.-H., Gemming, S. *GPU-based Atomistic Simulations on spatio-temporal experimental Scales* GTC, San Jose, CA, USA
10/2013	Kelling, J., Gemming, S. *Ab-initio investigation of carbides and of CNT junctions at finite temperature and under stress* NanoNet Workshop, Rossendorf, Germany
03/2013	Ódor, G., Schulz, H., Kelling, J., Heinig, K.-H., Máté Ferenc, N. *Extremely Large Scale Simulation of Surface Growth and Lattice Gases* GTC, San Jose, CA, USA
03/2013	Kelling, J., Heinig, K.-H. *Large Scale Atomistic Simulations on Nanostructure Evolution* DPG Sping Meeting, Sektion Kondensierte Materie, Dresden, Germany
10/2012	Kelling, J., Heinig, K.-H. *Large Scale Atomistic Simulations on Nanostructure Evolution* Workshop Response Treatment for the Dynamical Properties of Materials with the ABINIT Package, Zürich, Switzerland