Calculs numériques et modèles : études de cas pour les disques durs en physique statistique et pour l'apprentissage profond

Botao Li

PSL UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École normale supérieure
LPENS

Computation and models: Case studies for hard disks in statistical physics and for deep learning

Soutenue par
BOTAO LI

Le 14 décembre 2022

École doctorale nº 564
Physique en Île-de-France

Spécialité
Physique

Composition du jury :

Michael ENGEL FAU Erlangen-Nürnberg	Rapporteur
Ralf EVERAERS ENS de Lyon	Examinateur
Jan KIERFELD TU Dortmund	Rapporteur
Werner KRAUTH ENS	Directeur de thèse
Rodolphe VUILLEUMIER ENS	Président du jury
Zorana ZERAVCIC ESPCI Paris	Examinatrice

ENS | PSL

LPENS
LABORATOIRE DE PHYSIQUE
DE L'ÉCOLE NORMALE SUPÉRIEURE

Contents

Acknowledgement

First and foremost, I would like to express my sincere gratitude to Prof. Werner Krauth, my supervisor, for allowing me to carry out my doctoral study, guiding me through my research, and providing all kinds of support. I also would like to thank Prof. Thierry Mora for being my mentor. I thank Prof. Félix Werner for being my tutor and chatting with me frequently during the Covid time. I thank Prof. Michael Engel, Prof. Jan Kierfeld, Prof. Ralf Everaers, Prof. Rodolphe Vuilleumier, and Dr. Zorana Zeravcic for evaluating this work, especially Prof. Michael Engel and Prof. Jan Kierfeld for being my reporter.

My sincere thanks go to all my collaborators. I thank Prof. Anthony C. Maggs for the stimulating discussion and for providing me with computational resources during my doctoral study. I thank Dr. Liang Qin for the helpful discussion that allows me to start my research. I thank Philipp Höllmer for our discussion regarding sparse packings of hard disks and non-reversible Markov-chain Monte Carlo algorithms and for all the technical advice. I thank Dr. Yoshihiko Nishikawa for the discussion regarding GPU programming, state-of-the-art hard-disk pressure computation, and data analysis. I thank Dr. Etienne Bernard for generously sharing his fine-tuned program with us. I thank Prof. Synge Todo for the discussion of parallelizing the event-chain Monte Carlo algorithm. I thank Dr. Ze Lei for the discussion on the same topic. I also would like to thank Nicolas Noirault and Louis Carillo for the beautiful initial works during their internships that later turned into publications. I thank Dr. Riccardo Rossi and Dr. Fedor Šimkovic for sharing their idea of Many Configurations Markov Chain Monte Carlo. I thank Ziyin Liu for introducing me to the research of theoretical machine learning and initializing two interesting research projects. I also thank James B. Simon, Dr. Xiangming Meng, and Prof. Masahito Ueda for their contribution to these projects. It is a pleasure working with them.

I thank all the people that instructed or helped me in the past years; Prof. Yifu Cai and Prof. Yi Mao for introducing me to the research when I was an undergraduate at the university of science and technology of China; Dr. Yifan Chen for introducing me to the graduate opportunities in France and for all the advice for a newcomer to France; Prof. Stephane Munier and Prof. Pascal Paganini for allowing me to finish my high-energy physics master program at Ecole Polytechnique with a statistical physics project; Prof. Emilian Dudas, Prof. Christoph Kopper, Prof. François Gelis, Prof. Alain Aspect, Prof. Kirone Mallick, and many other professors for providing fascinating lectures during my master; Prof. Naomichi Hatano for guiding me in my very first research project in statistical physics and taking care of me during my visit in Japan; and of course Prof. Werner Krauth for his lectures on statistical physics, and Dr. Valentina Ros for the tutorial session after the lecture.

I would like to sincerely thank Prof. Jean-Marc Berroir, director of "Laboratoire de Physique de l'Ecole normale supérieure", especially for his help for my Ph.D. project. I thank Prof. Ly-

déric Bocquet equally for his essential help. I thank Prof. Jean-François Allemand and Prof. Frédéric Chevy, the two successive directors of the doctoral school, for their support that is beyond their duty. I thank administrative staff, in particular Olga Hodges, Hawa Traore, Christine Chambon, Laura Baron-Ledez, Fouzia Bouzid at the administration at ENS, and Mélanie Neguiral, Lucille Amard at CNRS for walking me through complicated administrative procedures. I equally thank the technical staff, in particular Yann Colin, for taking care of all problems I encountered regarding the internet and my computer. I cannot imagine my life in the department without your help.

In the second year of my Ph.D., I had the chance to teach at Université Paris Cité. I would like to thank Prof. Francesca Carosella for giving me this chance and my colleagues: Prof. Paolo Galatola, Prof. Yann Rasera, Prof. Giuliano Orso, Paul Jeammet, Prof. Renaud Belmont, and Thomas Richardson, for their guidance, help, and collaboration.

I would like to thank everyone that accompanied me during my Ph.D. Members of my group locate all around the world, and I enjoy their short and precious stay and, more importantly, their constant digital presence. I thank (for a second time) Prof. Werner Krauth, Prof. Anthony C. Maggs, Dr. Liang Qin, Dr. Philipp Höllmer, Dr. Yoshihiko Nishikawa, Dr. Ze Lei, Nicolas Noirault, Louis Carillo, and Shanglun Feng for sharing opinions and anecdotes about everything. I am grateful that I have friends with whom I can discuss both physics and life: Ziyin Liu, my friend since high school, provided me with crucial advice for almost every aspect of my life; Peng Cheng, my friend since undergraduate and my flatmate during my master, shared his insight into movies and news; Jiming Wu explained me the topic he is interested in physics, and his insight into foods; Jiaxin Qiao and Yi Zhang gave me a lot of practical advice. I would like to thank everyone that has shared the office with me: Dr. Liang Qin, who toured me around ENS when I first arrived; Shanglun Feng, who brought me the latest news about the university where I did my undergraduate; Alwin Philippe, who has a keen interest in different cultures; Gauthier Mukerjee, who decorated our office. Also, I would like to thank Dr. Jiaxin Qiao, Dr. Cathelijne ter Burg, Arnaud Bigué, Dr. Antoine Bourget, and Dr. Anxo Farina-Biasi. I also want to thank all the Ph.D. students and postdocs that I chatted with during the Ph.D. and postdoc meetings, in particular, Mariia Legenkaia. I also would like to thank my friends, in particular Rui Pang, Yifan Zhao, Huan Wu, Hanyuan Peng, and Shuyang Li, for all the support I received during my daily life. I would like to sincerely thank my parents for their support throughout my studies.

Last but not least, I thank Prof. Werner Krauth, Philipp Höllmer, Dr. Yoshihiko Nishikawa, Prof. Anthony C. Maggs, and Ziyin Liu for reading my thesis. Especially, I thank Prof. Werner Krauth for the time and energy that he spent in rereading this manuscript and on the procedure of my defense.

Chapter 1

Introduction

Physics describes the structure of matter and the interactions between the fundamental constituents of the observable universe [19] using mathematical language [7]. Specifically, the physical world is represented through mathematical models that are constructed considering both the resemblance of reality and the insight into nature. Very often, models are constructed to explain a given phenomenon, but are then used to predict other phenomena in the physical world. However, mathematical models of physical reality can be very difficult to "solve" and even only to "understand". To explain what we refer to as a solution, Onsager famously derived the expression for the partition function of the two-dimensional Ising model, thereby granting access to quantities related to the equilibrium phase transition [95]. The fact that such a phase transition must exist had already been proven (that is, "understood") by Peierls using an indirect argument [98]. In physics, models can be exactly solved in the above sense only in exceptional circumstances [9], and it may be even very difficult to arrive at a certain level of understanding. In this thesis, we examine the role of computation (in particular, stochastic computation) to arrive at a level of understanding that approaches the solution of models.

Stochastic computation dates back to 1777, when the French naturalist Buffon imagined the needle-throwing experiment in order to estimate the value of the number π [62]. Still, it was the invention of the computer that marks the practical beginning of stochastic computation in the Sciences,[1] in shape of Monte Carlo methods that date back to the 1940s [85]. Markov-chain Monte Carlo (MCMC) and molecular dynamics (MD), the numerical solution of Newton's equations on the computer, originated in the 1950s. Both methods rely on the concept of sampling. In this context, the use of the computer creates a number of problems. First, programs are complicated. Although a program can be formally verified, the vast majority of programs is not. Traditionally, the implementation detail of programs is also not public. In this thesis, we have been directly in contact with both problems. On the one hand, we have formally verified a multithreaded program whose behavior is erratic. On the other hand, we have led a consistent effort to publish our computer programs alongside our publications.

Second, there is randomness in stochastic computation. On a deterministic machine, creating true randomness is fundamentally impossible. This has posed a serious problem

[1] Computers may also produce symbolic output, as for example in computer algebra [126]. However, this mere extension of analytical derivations is not the focus of this thesis.

decades ago [61], but is considered less relevant now. Still, the output of the program remains fundamentally stochastic and hard to interpret. In this thesis, we have worked on the analysis of stochastic outputs from the programs. The randomness in Markov-chain Monte Carlo and molecular dynamics changes over time, and the correctness of the output is guaranteed only in the infinite-time limit. This is a fundamental and complicated problem which we are also in direct contact with in this thesis.

In this thesis, we study the computation for the hard-disk model, perceived by us as an example of sampling-based computation in general. From our experience, we break down this computation (for the hard-disk model but we believe also in general) into multiple aspects with a stair-case structure as shown in Fig. 1.1. The first level is the sampling algorithm, represented by a Markov-chain Monte Carlo or a molecular-dynamics algorithm. Algorithms feature vastly differing convergence times after which their sample outputs can be used. In this thesis, we have investigated various Markov-chain Monte Carlo algorithms and the molecular-dynamics algorithm for sampling hard-disk configurations. We have contributed to the algorithms by developing a multithreaded Markov-chain Monte Carlo algorithm. Also, we have studied a fundamental aspect of the algorithms, namely their convergence time, in specific circumstances. The second level is that of the observable. In the hard-disk model, one of the simplest observable is the pressure. We have contributed to a better understanding of the pressure by deriving various pressure estimators from its definition and reviewing the historical pressure computations. The third level is the implementation. The implementation is to produce computer programs for algorithms. In this thesis, we have implemented the Markov-chain Monte Carlo algorithms in Python, Go, C++, and CUDA. The programs run on a variety of hardware, including singlethread CPU, multithread shared memory CPU of both x86 and ARM architecture, and GPUs that allow for massive parallelization. We have formally verified our parallel implementation of a Markov-chain Monte Carlo in difficult circumstances by mapping the program onto an absorbing Markov chain. The fourth level in Fig. 1.1 is the statistical analysis of the output of the program. As an example, computing error bars for pressure estimates is difficult due to the correlation in the output. The final level is the interpretation of results of the statistical analysis, that is, to give meaning to the numbers and to draw conclusions relevant for physics. The phase behavior of the hard-disk model is itself complicated, and its behavior varies with respect to system sizes. Even if all of the previous steps are performed correctly, wrong qualitative conclusions can be drawn if the result of the computations is not correctly interpreted.

Not all computation problems require exactness as in the hard-disk model. Oftentimes, an efficient approximation can be useful. The structural convergence problem of the protein, a long-standing problem for thermodynamic or kinetic simulation of protein physics [18], has recently been tackled by AlphaFold [54]. AlphaFold is a model based on an artificial neural network that predicts the structure of the protein, and is an example of deep learning, a subject that we have also studied during the thesis. Artificial neural networks can approximate any function [47], and the use of multiple-layer neural network is generally referred to as "deep learning". Unlike sampling algorithms that only consist of explicit operations, the artificial neural network is a black box. Furthermore, the mechanism behind the performance of neural network is not well understood. This motivates our analysis simplified deep-learning models and optimization algorithms from a theoretical perspective in this thesis.

In the remainder of this introductory chapter, we summarize and motivate the five publications that have originated in this thesis. In the subsequent chapters, we synthesize the contents in the publications according to the stair-case structure.[2] In chapter 2, we provide an introduction to the hard-disk model from both a kinematic and a statistical-physics point of view. Moreover, we introduce sparse packings of hard disks, which play a crucial role in understanding and benchmarking the sampling algorithms. In chapter 3, we survey the hard-disk sampling algorithms, and discuss the correctness criteria of Markov-chain Monte Carlo algorithms, that is, the balance conditions, the irreducibility and the aperiodicity. We also describe our multithreaded sampling algorithm, published in [71]. Furthermore, we study the convergence time of the algorithms and their relation to sparse packings. The relevant results are published in [46]. In chapter 4, we introduce the observables for the hard-disk model. Notably, we describe the derivation of different hard-disk pressure estimators, published in [70]. In chapter 5, we discuss the implementation of algorithms. Particularly, we consider the formal verification and software tests for the computer programs and present the current state of the performance of programs. In chapter 6, we list the statistical analyses involved in the pressure computation of hard-disk model. In chapter 7, we interpret the outputs of the pressure computation from the computation point of view. We also discuss the historic results, and their interpretation in their epoch and today. The relevant content is the subject of the manuscript [70].

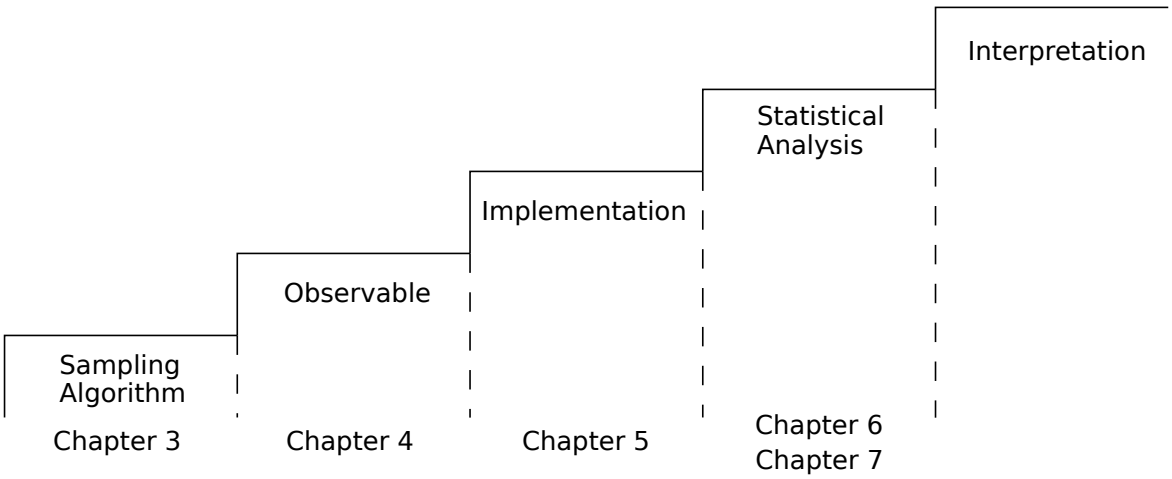

Fig. 1.1 *The stair-case structure of sampling-based computations for the hard-disk model, and the correspondence between the steps and chapters.*

1.1 Multithreaded event-chain Monte Carlo with local times

In this subsection, we motivate and summarize our first publication, which presents the first multithreaded (parallel) program for the event-chain Monte Carlo algorithm.

The history of the hard-disk model dates back to Daniel Bernoulli [13], and computer calculations have been performed since the 1950s [84]. As will be discussed throughout

[2]The publications are attached at the end of the thesis. Hyperlinks throughout the text connect the text to relevant contents in the attached publications in order to avoid duplication.

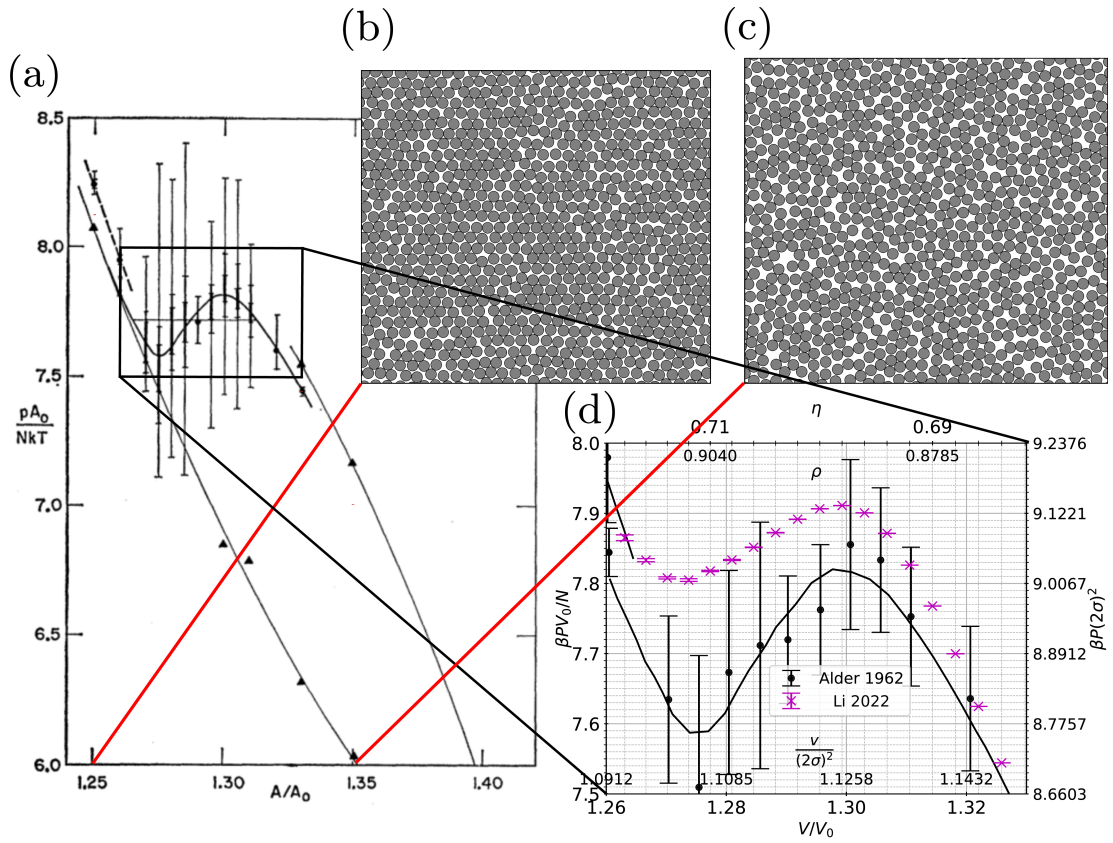

Fig. 1.2 *Hard-disk pressure computations.* (*a*) *The original equation of state computed by Alder and Wainwright in 1962, figure taken from* [2]. (*b*) *The hard-disk model at high density, demonstrating a rather ordered structure.* (*c*) *The hard-disk model at low density, having a disordered look compared with the high-density one.* (*d*) *The comparison of Alder and Wainwright's original result with the modern reproduction of the same computation.* (*Figure from* [70].)

this thesis, the model has the simplest possible short-range interactions: no two disks can overlap (see chapter 2 for a discussion). When disks are not overlapping, they are equivalent to ideal gas particles. Snapshots of the hard-disk model are shown in Fig. 1.2(b)(c).

Although the interaction between individual hard disks is easy to understand, the hard-disk model shows complicated collective behavior and, most interestingly, two phase transitions. Even today, our understanding of these phase transitions relies heavily on simulations, and no qualitative results corresponding to Onsager's and Peierls's work in the Ising model has been produced. We believe that progress in our understanding of the hard-disk model will continue to depend on the further development of algorithms.

The simulation of the hard disks is done by three algorithms, the event-chain Monte Carlo algorithm [11], massively parallel Monte Carlo algorithms [5], and modern event-driven molecular dynamics [49]. Both the massively parallel Monte Carlo algorithm and event-chain Monte Carlo algorithm outperform the Metropolis algorithm, the first Markov-chain Monte Carlo algorithm for sampling hard-disk configurations, by orders of magni-

tudes. The massively parallel Monte Carlo algorithm benefit from massively parallel hardware, while the event-chain Monte Carlo algorithm benefits from the efficiency of each move.

Similar to molecular-dynamics calculations of hard disks, the event-chain Monte Carlo algorithm is event-driven. The parallelization of event-driven molecular dynamics is notoriously hard since the causality can be easily broken. An event-driven algorithm computes the position of the disks at each collision, and the times between collisions are in general different. As a consequence, a given disk can be ahead of the other disks in time, and its collisions are computed using the out-of-date trajectories of the other disks. The event-driven molecular-dynamics algorithm has not been successfully parallelized. In contrast, we were able to show in [71], that the event-chain Monte Carlo algorithm can be consistently and efficiently parallelized, using an inherently different approach than previous attempts based on domain decomposition [56, 57]. Here, [57] proposed the key insight: "The primary obstacle in parallelizing event-chain Monte Carlo consists in preserving the correct causal relations between subsequent chains." In [71], we resolve the causality issue by introducing a "local time" for each disk and a "horizon condition" that is checked for each potential collision. Checking the horizon condition is computational possible even for large systems, since the number of potential collisions is less than three for each moving disk. The details of the algorithm are discussed in chapter 3.

Besides the algorithmic problem, the parallelization of event-chain Monte Carlo algorithm is technically challenging because of the computer architecture. Our algorithm has to be implemented on a multithreaded shared-memory machine. Multiple independent computers work in the very same system in memory. These "independent computers" are referred to as threads, each of which moves a disk. Multiple moving disks may collide with the same disks, and multiple threads may access the same place in memory. Having multiple threads writing into the same place in memory, referred to as data racing, must be avoided. In computer programs, this problem is commonly solved by locks—a thread locks some memory location, thus forbidding writing to this place by other threads. However, the implementation of locks greatly hurts the performance of the program. In [71], we have solved the data racing problem while maintaining a superior performance by using lock-free programming featuring atomic operations [14].

Furthermore, multithreaded computer programs are difficult to validate. Each of the threads runs at its own pace, and the order of operations on different threads cannot be controlled. This creates a type of "implementation randomness" that is different from the randomness in the stochastic computation. The correct output of the program has to be irrelevant to the implementation randomness. We rigorously prove the correctness of our program by formal verification using the framework of sequential consistency proposed by Lamport [65]. This is an example of how the issues related to the "implementation" level in the stair-case can be resolved (see Fig. 1.1).

1.2 Sparse hard-disk packings and local Markov chains

In this subsection, we motivate and summarize our second publication, which presents the sparse packing of hard disks and their relation to the Markov-chain Monte Carlo algorithms.

The optimal packing of hard disks in 2D is a triangular lattice of packing fraction $\frac{\pi}{2\sqrt{3}}$, in which not a single disk can be moved. At low density, there are less-known jammed

configurations in which no infinitesimal single-disk moves are allowed. One exemplary sparse packing of hard disks is the Böröczky packing, constructed first by Böröczky [16] and later also discussed by Kahle [55]. In the Böröczky packing, each disk is blocked by at least three of its neighbors. The Böröczky packing is only locally stable and not collectively stable, as collective moves of the disks can break it. In [46], we proposed it as a model for creating a bottleneck in Markov-chain Monte Carlo algorithms.

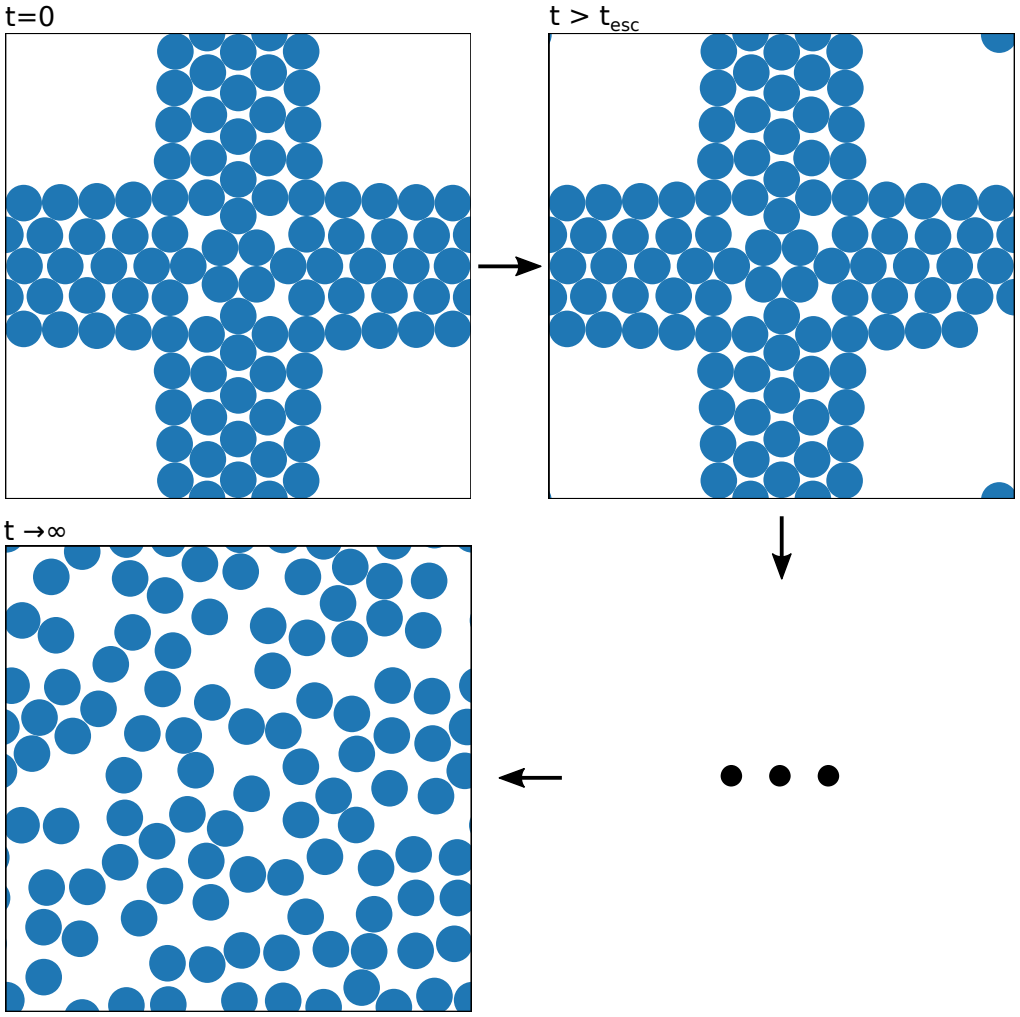

Fig. 1.3 *Escaping the ε-relaxed Böröczky configuration with $\epsilon = 10^{-10}$. The initial configuration is a Böröczky packing with disks shrunk by a factor of $1 - \epsilon$. After t_{esc}, a disk breaks free, but the structure may still be intact. In the limit $t \to \infty$, the structure originated from Böröczky packing is broken, and typical configurations are sampled. For the variants of event-chain Monte Carlo algorithms, t_{esc} has different scalings. The time it takes to escape the ε-relaxed Böröczky configuration serves as an indicator of the performance of the algorithm.*

In [46], we start simulation of the Metropolis algorithm, as well as the (straight) event-chain Monte Carlo algorithm, and its reflective [11], forward [87], and Newtonian [60] variants from ε-relaxed Böröczky configurations, the Böröczky packing after shrinking the disk

radii by $1-\epsilon$. We then measure the t_{esc}, the number of steps it takes for a single disk to escape the structure is measured. As ϵ approaches 0, the escape time t_{esc} of different Markov-chain Monte Carlo algorithms diverges with different scalings with respect to ϵ. In the limit of $t \to \infty$, the structure originated from Böröczky packing disappears completely. Fig. 1.3 shows how the structure disassembles. The ε-relaxed Böröczky configuration can be seen as a bottleneck of the sampling algorithm, and it serves as a benchmark of the algorithms, a useful tool in the first level of the stair-case (see Fig. 1.1).

Although no event-chain Monte Carlo algorithm can escape the (non-relaxed)Böröczky packing, in [46] we discuss why this is not relevant for the practical use of the algorithms. The infinite escape time indicates infinite pressure, which is not the case of interest in physics. Escaping the Böröczky packing is impossible in an NVT simulation, but relatively easy in an NPT simulation [62, 128]. Also, the sample space of the hard-disk model is high-dimensional, and its connectivity is likely not affected by Böröczky packings.

1.3 Hard-disk computer simulations—a historic perspective

In this subsection, we motivate and summarize our third publication, which discusses the computation of pressure in the hard-disk model.

Pressure is a priori a simple (that is, unambiguous) observable that has been computed since the 1950s. In fact, Metropolis et al. calculated the pressure using a MCMC algorithm and a polynomial extrapolation [84]. Several years later, Alder and Wainwright computed pressure in event-driven molecular dynamics simulations using the collision rate [2]. After these first results, pressure was computed for larger and larger systems [10, 29, 50, 51, 78, 102, 135]. Astonishingly, despite new results appearing, the values of the pressure (that is, of the equation of state) long remained contradictory. The discussion of the physical properties of the hard-disk model concentrated on the behavior of correlation functions, but in the absence of converged pressures, they were themselves biased.

In this thesis, we try to solve the pressure-computation problem from multiple perspectives. We have derived pressure estimators from the two definitions of pressure. These exact estimators coincide even for finite systems. They are then used to compute the pressure to high precision. We also trial the statistical analysis of computing the error bar of pressure. By using state-of-the-art sampling algorithms, exact estimator formulas, and a reliable statistical analysis, we compute the pressure to more than five digits, which can be used as a reference for potential future works.

Our high-precision results are published together with the digitized historic results in an open-source program, and can be compared visually by plotting a synopsis figure. Our database contains the pressure values and their error bars taken from the literature [2, 10, 29, 51, 52, 78, 84, 102, 135]. Some of the historic computations are reproduced, and the modern reproduction is compared with the original results. The results are presented in [70]. This paper addresses two aspects of the stair-case structure in Fig. 1.1, namely the observable and the statistical analysis. We succeeded in obtaining, after decades, reliable five-digit precision estimates of the pressure. We believe our estimates can last decades, just like the works by Onsager and Peierls for the Ising model, although, of course, this cannot be mathematically proven.

1.4 Simplified models and optimization algorithms in deep learning

In this subsection, we motivate and summarize our works related to deep learning, contained in the fourth publication and the fifth publication. In particular, in the fourth publication, we study the stochastic gradient descent algorithm in simple and fine-tuned models, and in the fifth publication we find the exact expression for the minima of the loss function of a general deep linear network. Either the fourth publication or the fifth publication has its own context and has appendices containing all the detailed discussion. Unlike the publications studying the hard-disk computation, their contents are not discussed separately in this thesis.

Deep learning allows for computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction [68], which finds its use in various fields, as for example in natural language processing [15, 22, 125], computer vision [32, 64], and optimal control [89, 90]. Deep learning is also implemented for the problems used to be solved by simulations (Alphafold [54] as discussed previously). The generative models are designed with sampling problems in mind [113], serving the same purpose as the Markov-chain Monte Carlo algorithms. However, not all computational tasks require exactness. Also, the importance sampling technique, widely used in MCMC, benefits from deep learning. A deep learning model can approximate the probability distribution of the proposed move and boost the performance of the MCMC algorithm [38, 92]. There are a large number of models in deep learning [68, 110]. In this thesis, we analyze deep learning in restrictive settings, limiting ourselves to the prediction problem using a fully connected neural network. Nevertheless, we believe that the conclusion of our studies apply to deep learning in general.

The fully connected neural network of Fig. 1.4 has a simple structure compared to other models. The model can be divided into multiple layers, whose output is the input of the next layer. Nodes in any layer connect to all nodes in the previous and next layer. Each connection is associated with a parameter. The first layer is the input, represented by a (high-dimensional) vector in which each component is viewed as a node. The value at each node of the input layer is determined by first summing over all of the values at the connected nodes in the previous layer, each multiplied by the parameter associated with the connection, then inputting the result of the sum into an (usually non-linear) activation function. The final layer is the output layer, and the in-between layers are referred to as hidden layers. The number of layer is referred to as depth, and the number of nodes in a layer is referred to as the width of that layer.

We study deep learning in the settings of a prediction problem, one of the simplest problems that can be solved by deep learning. The prediction problem is presented as following: given a collection of pairs consisting of a vector input and a scalar output as training data, predict the output of an input absent in the training data. This problem is solved by tuning the parameters in the neural network such that the relation between the input and output in the training data is approximated by the neural network. The neural network is a universal approximator [23, 47]. A deep neural network approximately describes any relation. The parameters are tuned by minimizing the difference between the predicted output computed from the input vector and the output in the training data, expressed by the mean square error. Finding the optimal parameter, referred to as training in the context of machine learn-

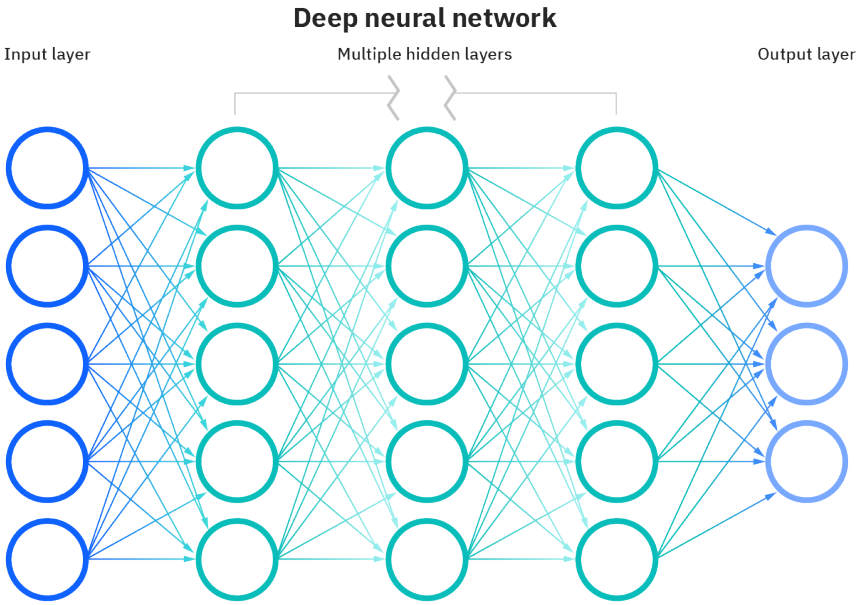

Fig. 1.4 *A visualization of the structure of a fully connected deep neural network. The each arrow corresponds to a weight. The arrow comes out of a node carries the value of the node multiplied by the weight. Multiple arrow going into a node indicates the value of the node depends on the sum of all of the values carried by the arrows. The value of the node is given by the activation function using the sum as input. We study the minima of the loss function of this model when the activation function is linear.* (*Figure from https://www.ibm.com/cloud/learn/neural-networks.*)

ing, is thus an optimization problem of minimizing the mean square error, referred to as the training loss in the same context. Usually the sum of squared parameters are added into the loss function to prevent the parameters from growing indefinitely, and this term is referred to as regularization. The performance of a trained network is evaluated by how accurate the prediction is, which is expressed as the least square error evaluated in the test data, a data set that shares no data point with the training data. The neural network may learn the undesired relation unique to the training data, resulting in small training loss but large test loss. Such neural networks, or such parameters in the neural network, are referred to as being unable to generalize.

The loss function, due to its complicated dependence on the parameters, has a large number of maxima and minima [58, 116, 123]. The loss function is thus referred to as a landscape. An example of a loss landscape is shown in Fig. 1.5. The optimal choice of the parameters must lie in one of the minima. However, it is believed that the weights in different minima of similar loss value do not describe the relation between input and output equally well. Practical observation shows that when the training loss already reaches its minimum, continuing to train improves generalization [93]. Thus, finding the optimal parameters requires asking the question of which minimum generalizes the best.

Identifying the best minimum is not the end of the story. In practice, reaching the best

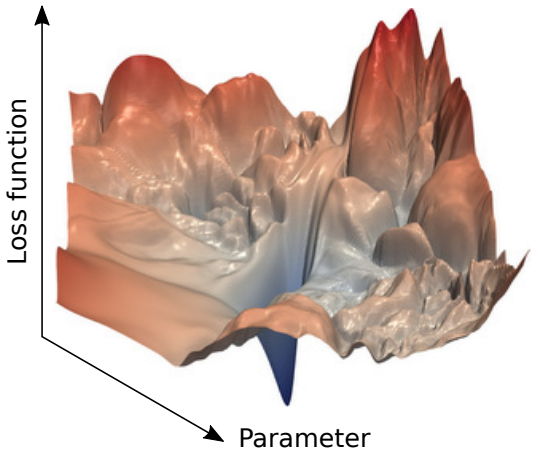

Fig. 1.5 *A visualization of the loss function (landscape) of neural networks. The loss function has a large number of local maxima, local minima, and saddle points. The goal of training in deep learning is finding a minimum that generalizes well. This is a complicated optimization problem. Some of the local minima in the landscape generalize badly and are not desired. Also, the optimization algorithm may converge to a local maximum in fine tuned cases. (Figure from* [72].)

minimum is not an easy task. In deep learning, the optimization is based on gradient calculation. In the gradient descent algorithm (GD), the parameters are updated by moving along the gradient in each iteration. The size of the displacement is controlled by a parameter called the learning rate. In the more popular stochastic gradient descent algorithm (SGD) [108, 109], the gradient is calculated using a fraction sampled from all the training data, and can be perceived as a noisy gradient. The convergence of SGD requires the learning rate to decrease gradually [111]. To further boost the performance, one can take advantage of the stochastic gradient calculated in previous iterations and use an adaptive learning rate, resulting in the Adam [59] and AMSGrad [105] method. Regardless of which minimum is the best, the optimization algorithm must be able to escape local maxima and saddle points. Escaping a saddle point and a local maximum is guaranteed, only with assumptions for the loss function and noise in SGD [83, 99, 118].

The neural network is usually considered a black box. The dimensionality of the input is generally huge, and the model is too complicated, rendering it impossible to interpret how the output depends on the input. The loss function lives in a high-dimensional space, and the gradient is a high-dimensional vector. The training data contains noise, leading to random training loss. Thus, understanding a realistic model is almost impossible. In response to this problem, we propose artificial data sets and simplified models whose behavior remains controlled. We study the loss landscape and behavior of optimization algorithms.

Firstly, we try to understand the loss landscape of a deep neural network. The simplified model we have chosen is a deep linear network [21] with dropout [115] and regularization. The deep linear network has the same structure as a deep neural network, but all activation functions are linear. We have found the closed-form expression for the minima in the training loss of the deep linear network. We first find the minima of the loss function for a two-layer linear network. The expression of the minimum is found by identifying the redun-

dant degree of freedom in the parameters in a loss function without regularization, which is also known as gauge symmetry in the context of physics. The regularization term then enforces gauge fixing and reduces the effective dimension of the parameters to 1. This solution implies that when increasing the strength of the regularization, the global minimum approaches the origin gradually, and stays there if the regularization strength is larger than a specific value calculated from the training data. This process is similar to a continuous phase transition [66]. Then, the expression of the minima is generalized to an arbitrarily deep linear network by induction. From the solution we draw the conclusion that the origin can be a local minimum, and initializing the weights near the origin may not be a good idea in deep learning.

We also study the optimization algorithms, notably the SGD, also referred to as the Robbins-Monroe algorithm [108, 109]. Understanding SGD is believed to be a crucial step in deep learning [4, 17, 26, 34, 37, 39, 73, 91, 123, 124, 132, 133, 134, 136]. It is generally difficult not only because the landscape is complicated, but also that the noise is state-dependent [134]. A large number of prior works studied how SGD escapes saddle point, but they take place in a restricted setting [36, 83, 99, 118]. To better understand SGD, we test these results in models containing one or two parameters and having an erratic loss function. In particular, we explicitly examine the behavior of SGD in a one-dimensional problem with a non-convex loss function containing two local minima and a local maximum. We model the fluctuation in SGD as a multiplicative noise [44]. We allow the SGD noise to vanish at the local maximum, which is not allowed in relevant works to our knowledge [36, 83, 99, 118]. Also, we assume that the learning rate of SGD is constant. Our simple setting allows us to calculate analytical how the SGD behaves. We also study the same problem in the continuous-time limit by solving a Fokker-Planck equation [103] to cross-check with our discrete-time results.

In our setting, we have observed that the SGD could converge to a local maximum when two of the crucial assumptions are broken: we assume that the noise is allowed to vanish, and the learning rate is kept constant. Our observation confirms the established results [36, 83, 99, 118], emphasizing the condition needed for SGD to escape a saddle point or a local maximum. The Adam and AMDGrad are examined numerically, and they can also converge to a local maximum. We also test SGD against the opinion that SGD has good generalization because it prefers a flat minimum, and that the flat minima generalizes better than sharp ones [42, 74, 81, 91, 112, 124, 131]. We show both analytically and numerically that the SGD, with specific noise, can converge to a sharp minimum. Throughout our investigation of SGD, we notice that the noise in SGD is no less important than the loss function.

Chapter 2

Hard disks in statistical physics

The hard-disk model is one of the simplest models in physics. It has a simple interaction and is specified, in the thermodynamic limit, by a single parameter, namely the density. Although the interaction of hard disks is repulsive, an effective attractive entropy-driven short-range force emerges from depletion [62].

There are only a few analytical results for the hard-disk model. The model is rigorously fluid at low density, and the virial expansion, which is proven to converge, is good enough to describe the model [41, 67]. The crystalline order is formed at the close packing [33]. Such order does not exist at any density lower than the fully-packed density as it is broken by large-scale fluctuations [106, 107]. At intermediate density, especially at the transition density, nothing holds rigorously even though insightful phase-transition scenarios are proposed [40]. Computation thus serves as the principal tool for investigating the hard-disk model.

In this chapter, we introduce to the hard-disk model. In section 2.1, we define the model and give an overview of its properties. In section 2.2, we describe sparse packings of hard disks, that is, non-fully-packed hard-disk configurations forbidding infinitesimal moves of independent disks. This chapter follows closely the introductory section II in the attached publication 3 [70], and the introductory section 2 in the attached publication 2 [46]. These publications then go on to discuss, on the one hand, pressure computation in detail, and on the other hand, benchmark Markov-chain Monte Carlo algorithms using the configurations derived from the sparse packing, which are also discussed in the chapter 4 and chapter 3 in this thesis.

2.1 Hard-disk model

In this section, we introduce the interaction and statistical ensembles of the hard-disk model. Then, we discuss its physical properties in statistical physics.

2.1.1 Definition of hard-disk model

The hard-disk model consists of N classical impenetrable two-dimensional disks of radius σ in a box of sides L_x, L_y and volume of $V = L_x L_y$, either with hard walls or periodic boundary conditions. The disk i is described by the coordinates of its center $\mathbf{x}_i = (x_i, y_i)$, where $i =$

$1, 2, 3, ..., N$ is the index of the disk. The disks are hard, in the sense that their interaction is characterized by a hard-core potential. Each disk carries a velocity $\mathbf{v}_i$ if necessary.

2.1.1.1 Hard-disk dynamics

In the hard-disk model, each disk is described by two parameters: its mass m and radius σ. The mass of each disk is located at the very center of the disk such that the momentum of a disk is carried by its center. The interaction between the disks is described by the hard-core potential, taking the form

$$V(r) = \begin{cases} \infty, & r \leq 2\sigma; \\ 0, & r > 2\sigma, \end{cases} \tag{2.1}$$

where r denotes the distance between disk centers. The disks have no rotational degree of freedom.

When there are no collisions between the disks, they all move in straight lines according to

$$\mathbf{x}_i = \mathbf{x}_{i,c} + \mathbf{v}_i(t - t_C), \tag{2.2}$$

where t is the time, t_C is the time of the last collision, and $\mathbf{x}_{i,C}$ is the position of the disk i at time $t = t_C$. Collisions take place when the two disks are in contact with each other, namely $|\mathbf{x}_i - \mathbf{x}_j| = 2\sigma$. The velocities of disks involved in collisions are updated according to Newtonian dynamics.[1] Let (i, j) be the pair of disks at collision. It is convenient to express their velocities in the center-of-mass frame before collision as [62]

$$\begin{cases} \mathbf{v}_i^- = \mathbf{v}^{\parallel} + \mathbf{v}^{\perp}; \\ \mathbf{v}_j^- = -\mathbf{v}^{\parallel} - \mathbf{v}^{\perp}. \end{cases} \tag{2.3}$$

Define $\Delta\mathbf{x}_{ij}^{\min} = \mathbf{x}_i - \mathbf{x}_j$ to be the vector connecting two disks at contact. Then, $\mathbf{v}^{\parallel}$ is the component of the velocity of disk i aligning with $\Delta\mathbf{x}_{ij}^{\min}$, while $\mathbf{v}^{\perp}$ is the other component perpendicular to $\Delta\mathbf{x}_{ij}^{\min}$. The transferred momentum between disk i and j is $m|\Delta v_{\text{pair}}^{\perp}|$, where $|\Delta v_{\text{pair}}^{\perp}| = 2|\mathbf{v}^{\perp}|$. The velocities after the collision in the center-of-mass frame are [62]

$$\begin{cases} \mathbf{v}_i^+ = \mathbf{v}^{\parallel} - \mathbf{v}^{\perp}; \\ \mathbf{v}_j^+ = -\mathbf{v}^{\parallel} + \mathbf{v}^{\perp}. \end{cases} \tag{2.4}$$

A pair collision is illustrated in Fig. 2.1(a).

The disks can collide with the walls in a box with hard walls. Analogous to the pair collision, let $\mathbf{n}$ be the direction perpendicular to the wall involved in the collision. For a disk-wall collision involving disk i, the sign of the component of $\mathbf{v}_i$ in the direction of collision is inverted at the collision, namely $\mathbf{v}_i^- \cdot \mathbf{n} = -\mathbf{v}_i^+ \cdot \mathbf{n}$. A wall collision is shown in Fig. 2.1(b). The rules of the velocity update presented here also appear in molecular dynamics and Newtonian event-chain Monte Carlo algorithm, as further discussed in chapter 3.

[1] We assume that only two disks are involved in the collision, as the probability of observing multiple-disk collision is zero.

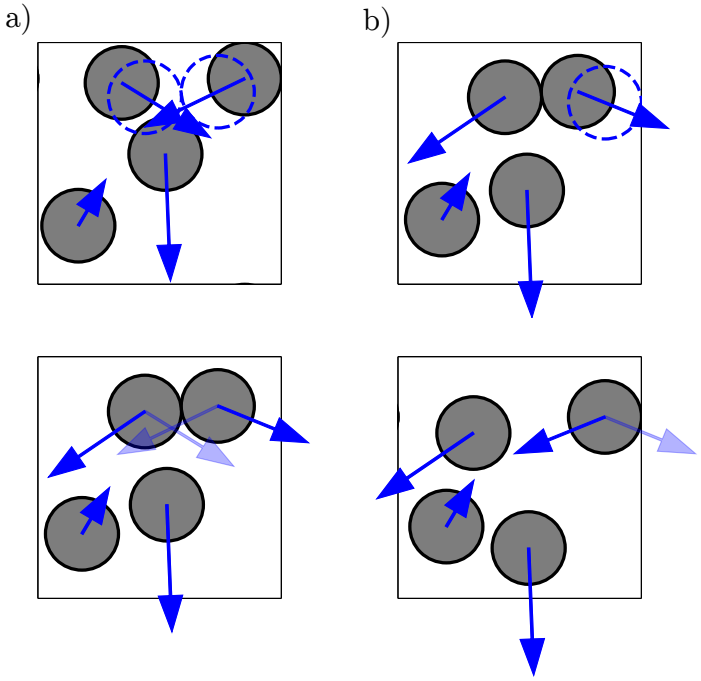

Fig. 2.1 *Illustrations of hard-disk collisions. The velocities of the disks are updated according to Newton's laws. The configurations in this figure are taken from an event-driven molecular dynamics run. In event-driven molecular dynamics, only the configurations at collisions are computed. (a) A collision between a pair of disks. (b) A collision between a disk and the wall.*

2.1.1.2 Statistical ensembles for the hard-disk model

Like the Ising model [48], the hard-disk model can be described by its configurations and their probability distribution. Each configuration is a $2N$-dimensional vector of positions of disks

$$\mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N). \tag{2.5}$$

In the hard-disk model, the total potential energy U, being the sum of the potential (2.1) between all pairs of hard disks, takes the form of

$$U(\mathbf{x}) = \begin{cases} \infty, & \text{if there are overlaps;} \\ 0, & \text{if there is no overlap,} \end{cases} \tag{2.6}$$

where an overlap between disk i and j is present if $|\mathbf{x}_i - \mathbf{x}_j| < 2\sigma$. We adopt the convention of setting the Boltzmann constant to 1. The statistical weight of the configurations is specified by the Boltzmann weight $\mu(\mathbf{x}) = \exp(-\beta U(\mathbf{x}))$, where $\beta = 1/T > 0$ is the inverse temperature. The weight of a configuration containing overlap is zero, and all configurations free of overlap have the same probability weight, that is, configurations with $\Theta(\mathbf{x}) = 1$, where $\Theta(\mathbf{x})$ is the member function of the set of legal configurations:

$$\Theta(\mathbf{x}) = \begin{cases} 1, & \forall (i,j), |\mathbf{x}_i - \mathbf{x}_j| > 2\sigma \\ 0, & \text{otherwise.} \end{cases} \tag{2.7}$$

The sample space, i.e. the space of configurations such that $\Theta(\mathbf{x}) = 1$, depends on the boundary conditions. For a non-periodic box with hard walls, disks cannot overlap with the walls. In a periodic system, the box is equivalent to a torus. The sample space for a non-periodic system is $\Omega_{\text{non-periodic}} = \text{supp}[\Theta] \cap [\sigma, L_x - \sigma]^N \times [\sigma, L_y - \sigma]^N$, and, for a system implementing the periodic boundary condition, it is $\Omega_{\text{periodic}} = \text{supp}[\Theta] \cap [0, L_x]^N \times [0, L_y]^N$. In a non-periodic box with hard walls, the partition function is defined such that

$$Z_{\text{non-periodic}}(V, \sigma) = \int_{\Omega_{\text{box}}} d\mu(\mathbf{x}). \tag{2.8}$$

For periodic boundary conditions, the partition function is defined as

$$Z_{\text{periodic}}(V, \sigma) = \int_{\Omega_{\text{periodic}}} d\mu(\mathbf{x}). \tag{2.9}$$

In principle, the partition function depends on the number of disks N, the disk radius σ, and the box geometry (L_x, L_y). As will be shown in section 2.1.2.3, the aspect ratio $\alpha = (L_x : L_y)$ is not a relevant parameter in the thermodynamic limit $N \to \infty$. Furthermore, the system can be rescaled, and σ can be set to $\frac{1}{2}$. In this case, the parameters of the model are further reduced to N and V. The hard-disk model is uniquely determined when N and V are specified. As will be discussed in section 2.1.2.1, the temperature T is not relevant for the hard-disk model. However, conventionally, this ensemble is still referred to as the NVT ensemble.

The free energy F of the hard-disk model is

$$F = -T \log(Z). \tag{2.10}$$

The pressure, namely the response to changing volume V, is calculated as

$$\beta P = -\frac{\partial F}{\partial V}. \tag{2.11}$$

The pressure will be further discussed in section 4.3, especially the different ways to change the volume in a finite system. Alternatively, one can treat the pressure P as the control parameter and the volume V as the response of changing P. Such an ensemble is referred to as the NPT ensemble. Although specific algorithms exist for hard-disk simulations in the NPT ensemble [62, 127, 129], simulations in the NPT ensemble are less efficient than in NVT ensemble. The NPT ensemble resolves problems posed by the sparse packings, as they corresponds to effectively infinite-pressure configurations (discussed in section 2.2). A detailed discussion about the Böröczky packing and the NPT ensemble can be found in section 4.2.2 in the attached publication 2 [46].

For large system sizes, the physical properties of the hard-disk model (as for any statistical-physics model with short-range interactions) are independent of N, and the volume V is the only relevant parameter. By rescaling, systems of $\sigma = \frac{1}{2}$ and different volumes can be mapped to systems of the same volume and different σ, that is, systems of different density. The volume is equivalent to the density of the system. The volume (density) has four equivalent expressions. The first is the ratio of the volume and fully-packed (minimal) volume $V_0 = 2\sqrt{3}N$. The second is the covering density η, that is, the area (volume) occupied by the disks compared to the area (volume) of the box. Third is the reduced density ρ, the number N of disks of radius $\frac{1}{2}$ divided by the volume and, finally, the inverse normalized density $\nu/(2\sigma)^2$ with $\nu = (2\sigma)^2/\rho$. Useful relations between these quantities are as follows:

$$\begin{aligned}
\frac{V}{V_0} &= \frac{\pi}{2\sqrt{3}}\frac{1}{\eta} = \frac{2}{\sqrt{3}}\frac{\nu}{(2\sigma)^2} \geq 1 \\
\eta &= \frac{\pi}{2\sqrt{3}}\frac{V_0}{V} = \frac{N}{V}\pi\sigma^2 \leq 0.907 \\
\frac{\nu}{(2\sigma)^2} &= \frac{\sqrt{3}}{2}\frac{V}{V_0} = \frac{\pi}{4}\frac{1}{\eta} = \frac{1}{\rho} \geq 0.866 \\
\rho &= \eta\frac{4}{\pi} = \frac{V_0}{V_0}\frac{2}{\sqrt{3}} = \frac{(2\sigma)^2}{\nu} \leq 1.155.
\end{aligned} \tag{2.12}$$

The historic pressure computations use a variety of units for the volume (density). We provide a program that allows comparing directly the data in various units, presented in appendix B.1 in the attached publication 3 [70].

In circumstances when velocities of hard disks have to be considered, for example in molecular dynamics, the hard-disk model is represented by the positions of hard disks together with their velocities, expressed by a tuple $(\mathbf{x}, \mathbf{v})$, where

$$\mathbf{v} = (\mathbf{v}_1, ..., \mathbf{v}_N). \tag{2.13}$$

is a $2N$ dimensional vector. As the velocity has its own space and probability distribution, the statistical ensemble of the hard-disk model contains a velocity space, and parameters N and V no longer fully determine the ensemble. In the NVT ensemble, the velocity space is $\mathbb{R}^{2N}$, and the probability distribution for the velocities is the Maxwell distribution

$p_{\text{Marwell}}(\mathbf{v}) \propto \exp\left(-\frac{1}{2}\beta m|\mathbf{v}|^2\right)$. In the NVE ensemble, in which the total energy E of the system is specified instead of the temperature T, the velocity is constrained on an energy shell defined by relation

$$E = \frac{1}{2}m|\mathbf{v}|^2, \tag{2.14}$$

and the probability density is the same for all configurations. There are ensembles that have even more constraints. For example, in the $NVE\mathbf{MR}$ ensemble of systems with periodic boundary conditions, the total momentum $\mathbf{M}$ and the center-of-mass position $\mathbf{R}$ is specified. Thus, the velocity space in the $NVE\mathbf{MR}$ ensemble is defined by the following conditions:

$$\sum_i v_{i_x} = \mathbf{M}_x/m, \tag{2.15}$$

$$\sum_i v_{i_y} = \mathbf{M}_y/m, \tag{2.16}$$

$$E = \frac{1}{2}m|\mathbf{v}|^2. \tag{2.17}$$

Event-driven molecular dynamics with periodic boundary condition and without a thermostat takes place in this ensemble [130]. Due to the additional conservation, the distribution for the velocity of a single disk differs in different ensembles, as will be discussed in section 2.1.2.1. Also, as will be discussed in chapter 4, the choice of the ensemble modifies the pressure and its estimators, but only in finite systems with periodic boundary conditions. As this thesis mainly concerns the Markov-chain Monte Carlo algorithm, the velocity in the hard-disk model is ignored in this thesis unless stated explicitly.

2.1.2 Properties of hard-disk model

The hard-disk model has many distinct properties. For example, the temperature can be factored out by the rescaling of time in the hard-disk model; the analytical distribution of the velocity is known; the phase-transition is driven by the depletion interaction; the aspect ratio plays an important role in finite systems, etc. This subsection, following closely section II in the attached publication 3 [70], introduces to these properties.

2.1.2.1 Basic properties

Rescaling all disk velocities by a factor α, the entire trajectory of hard-disk dynamics transforms as

$$\{\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)\} \xrightarrow{\mathbf{v}_i \to \alpha \mathbf{v}_i \forall i} \{\mathbf{x}_1(\tfrac{t}{\alpha}), \ldots, \mathbf{x}_N(\tfrac{t}{\alpha})\}. \tag{2.18}$$

The transformed trajectory is still the old one, but with a rescaled time. Since we are interested in the equilibrium properties of the hard-disk model, the time is irrelevant, which in turn indicates that the velocity is not relevant. Consequently, the hard-disk model is not sensitive to temperature. This property of hard-sphere models was already remarked by Daniel Bernoulli [13].

A fast disk collides more frequently with walls than a slow one. The probability distribution of the velocity perpendicular to a wall $v^{\perp}_{\text{wall}}$ at wall collisions is biased by a factor $|v^{\perp}_{\text{wall}}|$ with respect to the Maxwell distribution (for $N \to \infty$):

$$p(|v^{\perp}_{\text{wall}}|) \propto |v^{\perp}_{\text{wall}}| \exp\left(-\beta m (v^{\perp}_{\text{wall}})^2/2\right), \tag{2.19}$$

which has been described through the "Maxwell boundary condition" (see [62, Sect. 2.3.1]). For finite N, the distribution of the velocity perpendicular to a wall is derived from the surface element on the hypersphere of radius $R = \sqrt{v_1^2 + \cdots + v_n^2}$ (with $R^2 = 2N/(m\beta)$) in the NVE ensemble. Following the detailed derivation in II.A.1 in the attached publication 3 [70], one obtains

$$\left\langle \frac{1}{|v_{\text{wall}}^{\perp}|} \right\rangle = \frac{\sqrt{\pi}}{R} \frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \overset{N\to\infty}{\longrightarrow} \sqrt{\frac{\pi m\beta}{2}}, \tag{2.20a}$$

$$\left\langle |v_{\text{wall}}^{\perp}| \right\rangle = \frac{R\sqrt{\pi}}{2N} \frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \overset{N\to\infty}{\longrightarrow} \sqrt{\frac{\pi}{2m\beta}}, \tag{2.20b}$$

where in the limit of $N \to \infty$ the ratio of the Γ functions approaches $\sqrt{N}$) in the NVT ensemble.

The relative perpendicular velocities $\Delta v_{\text{pair}}^{\perp}$ is the component of the relative velocity $\mathbf{v}_i - \mathbf{v}_j$ perpendicular to the interface separating the disks i and j at their collision. A similar expression to (2.20) can also be derived for the probability distribution of relative perpendicular velocity $|\Delta v_{\text{pair}}^{\perp}|$. The positions are not correlated to the velocities, so the distribution of the velocities for the disks at a pair collision is exactly the same as for all the disks. The interface separating the disks at a collision can be viewed as a fictive wall, and the pair collision of the disks can be viewed as a collision with it. The isotropic velocity distribution guarantees that collisions with a fictive wall facing any direction is the same as the wall collision in the x or y direction, allowing one to rotate the coordinate system such that the x-axis after rotation is aligned with the line connecting the centers of the disks. We introduce the transformation

$$\mathbf{v} = (..., v_{i,x}, ..., v_{j,x}, ...) \to \mathbf{v}' = (..., (v_{i,x} + v_{j,x})/\sqrt{2}, ..., (v_{i,x} - v_{j,x})/\sqrt{2}, ...). \tag{2.21}$$

This transformation amounts to reparameterizing the velocity of two colliding disks using their relative velocity and their center-of-mass velocity. The probability weight of velocities is a function of $E = \frac{1}{2}m\sum_i |\mathbf{v}_i|^2$, and the transformation (2.21) leaves E unchanged. Thus, $p_{\text{Maxwell}}(\mathbf{v}) = p_{\text{Maxwell}}(\mathbf{v}')$, as the Jacobian of the transformation is 1. Thus, $\langle \Delta v_{\text{pair}}^{\perp} \rangle = \sqrt{2}\left\langle |v_{\text{wall}}^{\perp}| \right\rangle$, since calculating $\langle \Delta v_{\text{pair}}^{\perp} \rangle/\sqrt{2} = \left\langle (v_{i,x} - v_{j,x})/\sqrt{2} \right\rangle$ follows exactly the same procedure as deriving $\langle |v_{\text{wall}}^{\perp}| \rangle$. The same idea also applies for $\langle 1/\Delta v_{\text{pair}}^{\perp} \rangle$, and the mean values related to the relative perpendicular velocity is found by inserting a factor of $\sqrt{2}$ into (2.20):

$$\left\langle \frac{1}{|\Delta v_{\text{pair}}^{\perp}|} \right\rangle = \frac{\sqrt{2\pi}}{R} \frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \overset{N\to\infty}{\longrightarrow} \sqrt{\pi m\beta}, \tag{2.22a}$$

$$\left\langle |\Delta v_{\text{pair}}^{\perp}| \right\rangle = \frac{R\sqrt{\pi}}{\sqrt{2}N} \frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \overset{N\to\infty}{\longrightarrow} \sqrt{\frac{\pi}{m\beta}}. \tag{2.22b}$$

Equation (2.22) is used in chapter 4 to derive the pressure estimators.

In the NVE ensemble, the distribution of $v_{\text{wall}}^{\perp}$ has a probability density function $p_{v_{\text{wall}}^{\perp}}$ such that

$$p_{v_{\text{wall}}^{\perp}}(v_{\text{wall}}^{\perp}) \propto v_{\text{wall}}^{\perp}(1 - {v_{\text{wall}}^{\perp}}^2)^{\frac{2N-3}{2}}, \tag{2.23}$$

And consequently, the distribution function of $1/v_{\text{wall}}^{\perp}$ is given by

$$p_{1/v_{\text{wall}}^{\perp}}(x) \propto \frac{1}{x^3}\left(1-\frac{1}{x^2}\right)^{\frac{2N-3}{2}}, \tag{2.24}$$

where $x = 1/v_{\text{wall}}^{\perp}$ possesses a diverging variance. As will be discussed in chapter 4, some pressure estimators of molecular dynamics suffer from a diverging variance. In practice, they are substituted by equivalent estimators with finite variance after integrating over velocities.

2.1.2.2 Hard-disk thermodynamics and phase transition

In the hard-disk model, the Boltzmann weights are the same for all configurations, and disks not in contact do not directly sense each other. In consequence, the two possible fluid phases (namely the gas and the liquid) are confounded. A "depletion" interaction [6, 62] between disks nevertheless arises from the presence of other disks, effectively driving phase transitions. The three phases of the hard-disk model are solid (with long-range orientational correlations and an algebraic decay of positional correlations), hexatic (with algebraic decay of orientational correlation and exponential decay of positional correlation), and fluid (with all correlation functions exponentially decaying). The hexatic and solid phases have only been identified through numerical simulations. The system must be large enough to distinguish the hexatic and the solid phase.

In statistical mechanics, a homogeneous system is described by an equation of state—a quantity as a function of another. For the hard-disk model, they are conventionally the volume and the pressure. In the NVT ensemble, for a system possessing a first-order phase transition, the system may not be homogeneous for some volume. Two phases may coexist, separated by an interface with its own interface free energy.

The phase transition between the fluid and the hexatic phase in the hard-disk model is a first-order phase transition. In a periodic two-dimensional box for finite N, on increasing the density, a first-order phase transition first creates a bubble of the denser phase in the less dense phase (for hard disks: a hexatic bubble inside the fluid). An extra "Laplace" pressure is required to stabilize this bubble, rendering the overall pressure non-monotonous with V [80]. At larger density, the bubble of the minority phase grows to the size that is comparable to the rest of the system. The two phases arrange themselves such that there is a stripe winding around the periodic box. In this regime, the length of the two interfaces and thus the interface free energy do not change with V, so that the pressure remains approximately constant. Finally, for even larger density, a bubble of the fluid phase is formed in the surrounding hexatic phase (see Fig. 2.2). The phase coexistence through the first-order transition is specific to the NVT ensemble as certain specific volumes V/V_0 do not correspond to densities $\eta = (N/V)\pi\sigma^2$ of a homogenous stable phase for $N \to \infty$. Phase coexistence is absent in the NPT ensemble, and V/V_0 is discontinuous at the transition. The NPT ensemble thus has a simpler physical picture. However, when sampling in the NPT ensemble, exploring possible volumes is slow, rendering the NPT simulation less efficient than the NVT ones.

The phase coexistence and the non-monotonous equation of state are genuine equilibrium features at finite N. Moreover, the spatially inhomogeneous phase-separated equi-

librium state is reached from homogeneous initial configurations through a slow coarsening process, whose dynamics depends on the sampling algorithm. As will be discussed in chapter 3, this process can benchmark the sampling algorithms.

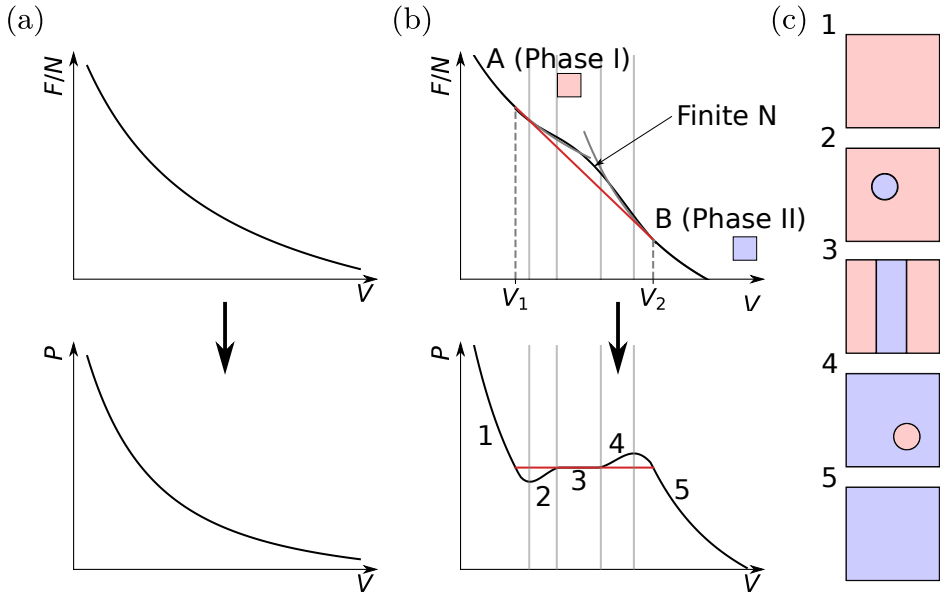

Fig. 2.2 *First-order phase transition in the NVT ensemble. (a) Free energy with increasing second derivative and a monotonously decreasing pressure. (b) Free energy with—for the infinite system—metastable branches starting at volumes V_1 and V_2, and a non-monotonous equilibrium pressure P for finite N. (c) Sequence of five regimes in a finite two-dimensional periodic box, with pure, "bubble" and "stripe" phases. (Figure from [70], see also [12].)*

2.1.2.3 Aspect ratio and N

For the hard disk model, it is easy to see that the aspect ratio plays an essential role in physics for a small N. As shown in Fig. 2.3, for the same V and $N = 72$, the disks align as a perfect crystal in the box with aspect ratio $\alpha = (9 : 8\sqrt{3}/2)$, but not in the box with aspect ratio $\alpha = (1 : \sqrt{3}/2)$. As a consequence, at density extremely close to being fully packed, the box with aspect ratio $\alpha = (9 : 8\sqrt{3}/2)$ can still host valid configurations, while in a box with aspect ratio $\alpha = (1 : \sqrt{3}/2)$ no configuration is allowed. This is an example of the boundary free energy in a finite system, whose share in total free energy vanishes in the thermodynamic limit.

In a perfect crystal, the hard disks are on a triangular lattice. Conventionally, the straight lines made up of N_x disks are aligned along the x direction, each of which is referred to as a row. The number of rows is then referred to as N_y, and $N = N_x N_y$. For a box hosting a fully packed configuration, the aspect ratio of the box is calculated by

$$\alpha = (N_x : N_y\sqrt{3}/2). \tag{2.25}$$

In the fully packed configuration, the parallelogram consisting of two rows appears repeatedly. Thus, N_y must be an even number in order to comply with the periodic boundary

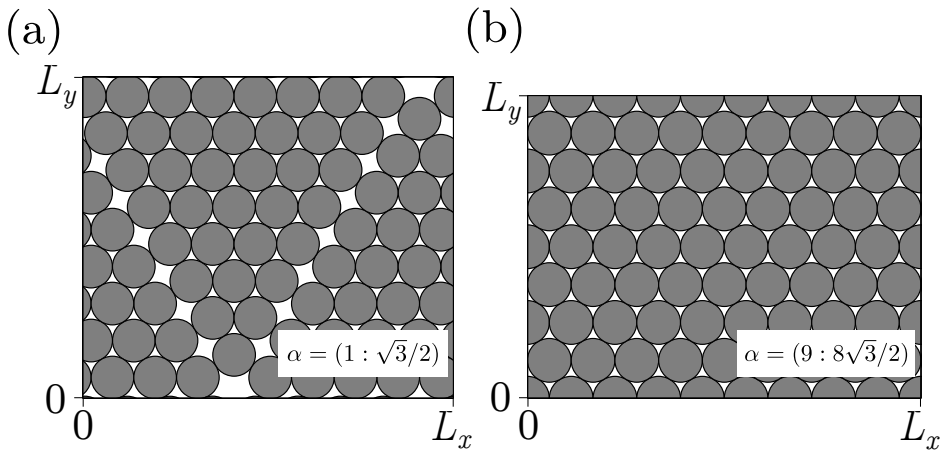

Fig. 2.3 *Packings for $N = 72$ disks for different aspect ratios α. (a) Periodic box with $\alpha = (1 : \sqrt{3}/2)$, with conjectured optimal packing. (b)Periodic box with $\alpha = (9 : 8\sqrt{3}/2)$ at the close-packing density $\eta = \pi/(2\sqrt{3})$. (Figure from [70].)*

conditions.

As will be discussed in section 7.1.1, a square box is preferred over the rectangular one. However, the square box does not host closed-packed configurations for small N. The idea value of N is obtained by finding a decomposition of $N = N_x N_y$ such that $\alpha = (N_x : N_y\sqrt{3}/2)$ is close to $(1 : 1)$. This is equivalent of finding a rational approximation of $\sqrt{3}/2 \approx 0.866$. The candidates are $6/7 = 0.8571, 7/8 = 0.875, 13/15 = 0.8667, 84/97 = 0.8660, ...$ Since each disk in a row takes up 2σ and each row takes up $\sqrt{3}\sigma$, in a square box, the number of rows has to be the larger than the number of disks per row, and it has to be even. The possible choices of (N_x, N_y) are limited to $(12, 14), (14, 16), (26, 30), (168, 194), ...$ If the upper limit of N is 500, the best choice is then $N_x = 14, N_y = 16$. Indeed, Metropolis et al. chose $N = 14 \times 16 = 224$ for computations in a square box.

2.2 Sparse packings

In this section, we introduce to sparse hard-disk packings discovered by Böröczky in 1964 [16]. They forbid any infinitesimal move of any individual disk and are conceptually and practically interesting in the context of the ergodicity of the sampling algorithms. In this section, we provide a general introduction to these packings and describe their construction, the consequences of their existence on the structure of sample space, and the collective moves that can break these configurations. The content in this section closely follows section 2 in the attached publication 2 [46]. Böröczky packings are closely related to fundamental aspects of Markov-chain Monte Carlo algorithms. The Böröczky packing, and the ε-relaxed Böröczky configuration that are derived from Böröczky packings, can be viewed as bottlenecks of the sampling algorithms. In particular, the ε-relaxed Böröczky configuration serves as a benchmark of the sampling algorithm and will be discussed in chapter 3. The behavior of the Markov-chain Monte Carlo algorithms in the ε-relaxed Böröczky configuration provides insight into how the algorithms perform in the case of difficult situations in more complicated models.

2.2.1 Definition

In the present section, we discuss Böröczky packings of N disks of radius $\sigma = 1$ in a periodic square box of sides L. The simulation box ranges from $-L/2$ to $L/2$ in both the x and the y direction. A Böröczky packing is locally stable, and each of its N disks is blocked, at a distance 2σ, by at least three other disks (taking into account periodic boundary conditions), with the contacts not all in the same half-plane. The opening angle of a disk i, the largest angle formed by the contacts to its neighbors, is then always smaller than π. In this thesis, we are interested in ECMC algorithms which move a single disk at a time in a continuous manner. Clearly, a locally stable packing cannot be escaped from through the infinitesimal single-disk moves of ECMC.

In a nutshell, Böröczky packings consist in cores and branches (as visible in Fig. 2.4). Böröczky packings can exist for different cores, and they depend on a bounding curve (more precisely: a convex polygonal chain) which encloses the branches, and which can be chosen more or less freely.

2.2.1.1 Construction of Böröczky packings

In the central simulation box, a finite-N Böröczky packing is built on a central core placed around $(0,0)$ (see the section 2.1.1 in the attached publication 2 [46] for a discussion of cores). Such cores were for example proposed in the original work by Böröczky [16] and by Kahle [55]. The positions of the disks in the core are considered constants and specified directly. This core connects to four periodic copies of the core centered at $(L,0)$, $(0,L)$, $(-L,0)$, and $(0,-L)$ by branches that have k separate layers (see section 2.1.2 in the attached publication 2 [46] for a detailed discussion of branches). The Böröczky packing is then constructed by completing the branches layer by layer. A Böröczky packing shares the symmetries of the central simulation box. Thus, only constructing the upper half of the right branches (the framed region in Fig. 2.4) is required. The other halves of the branches are constructed using symmetry. Depending on the y coordinate, the disks in the right upper half branch are labeled as A, B, and C disks. The positions of A disks are specified by a convex polygonal chain $\mathcal{A}$, possibly controlled by a parameter and found by trial and error. The position of B and C disks are determined iteratively. The position of each B disk is determined so that it is in contact with the A disk in its layer and the C disk in the previous layer, while the position of each C disk is determined so that it lies on the axis of symmetry and in contact with the B disk in the previous layer. The bounding curve is determined so that the branch approaches a crystal in the case of an infinite system and is able to join in the branch from the neighboring image system in other cases. For an infinite system, cores with different shapes, as for example that of a triangle, yield Böröczky packings in other geometries (see [16, 55] and [96, Section 9.3]).

2.2.2 Properties of Böröczky packings

The local stability of Böröczky packings only relies on the fact that all A disks lie on a largely arbitrary convex polygonal chain $\mathcal{A}$ [16]. The choice of $\mathcal{A}$ influences the qualitative properties of the packing. Although Böröczky packings are a big threat to ECMC algorithms, its dimensionality is is lower than that of sample space, and it is conjectured that they do not separate sample space into individual parts. Besides, if multiple disks move collectively,

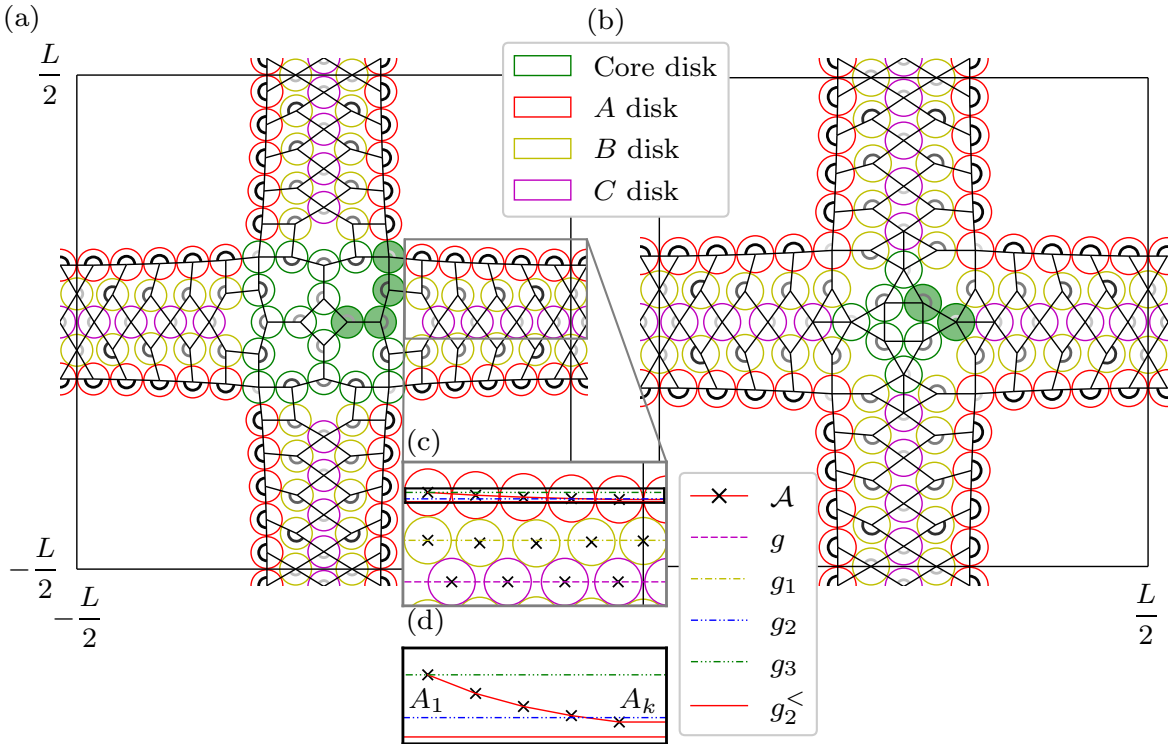

Fig. 2.4 *Hard-disk Böröczky packings, composed of a core and of four branches with $k = 5$ layers, with contact graphs and highlighted opening angles.* (*a*) *Packing with the Böröczky core* [16]. (*b*) *Packing with the Kahle core* [55]. (*c*) *Detail of a branch.* (*d*) *Convex polygonal chain $\mathcal{A}$, and referential horizontal lines $g_2^<$, g_2, and g_3 for creating the polygonal chain.* (*Figure from* [46].)

the Böröczky packing can be escaped. This is guaranteed by having fewer constraints than degrees of freedom. The escape modes can be found by performing a singular-value decomposition on the matrix representing the contacting disks. The existence of escape modes indicates that the collective-move local algorithms, such as molecular dynamics, may escape Böröczky packings. (The discussion of cluster Markov-chain Monte Carlo algorithms is beyond the scope of this thesis and hence not considered.) Nevertheless, the local moves in some Markov-chain Monte Carlo algorithms are arranged to achieve effective collective moves. This is observed by running the sampling algorithms on a Böröczky packing with slightly shrunk disks, introduced as ε-relaxed Böröczky configuration.

2.2.2.1 Contact graphs: local and collective stability

The contact graph of a Böröczky packing connects any two disks whose pair distance equals 2 (including periodic boundary conditions, see Fig. 2.4). In a Böröczky packing with $k \geq 1$ layers, the number N of disks and the number of contacts N_{contact} can be expressed as functions of k:

	N	N_{contact}
Böröczky core	$20k + 12$	$32k + 20$
Kahle core	$20k - 4$	$32k + 4$

. (2.26)

Thus, for all values of $k \geq 1$, $N-1 < N_{\text{contact}} < 2N-2$, indicating that the contact graph is not able to fix every degree of freedom in the system. This implies that collective displacements can escape from a Böröczky packing, which is thus not collectively stable [55]. However, the contacts efficiently block unidimensional moves. The consequence of this will be discussed in chapter 3.

2.2.2.2 Dimension of the space of Böröczky packings

As discussed in section 2.2.3 in the attached publication 2 [46], the Böröczky packing may exist for any density at large enough N. The dimension of space $\mathcal{B}$ of locally stable packings of N disks in a given box is lower than that of sample space. As discussed in section 2.2.2.1, the contacts in a Böröczky packing are described by a contact graph. Each independent edge in the contact graph decreases the dimensionality of Böröczky packings by one. In addition there is only a finite number (at most N^N) of contact graphs for a given N. Any configuration having disks in contact has infinite pressure. As the expectation of pressure is finite (except for the densest packing), the set of packings (and the configurations containing packings) must be of lower dimension. As the dimension of $\mathcal{B}$, for large N, is much lower than that of Ω, we conjecture $\Omega \setminus \mathcal{B}$ to be connected for a given η below the densest packing at large enough N although, in our understanding, this is proven only for a much lower density $\eta \sim 1/\sqrt{N}$ (see [8, 25]).

2.2.2.3 Escape modes

When all disks i, at positions $\mathbf{x}_i$, are moved to $\mathbf{x}_i + \boldsymbol{\Delta}_i$, the squared distance between two disks in contact changes from $|\mathbf{x}_i - \mathbf{x}_j|^2$ to

$$|\mathbf{x}_i + \boldsymbol{\Delta}_i - (\mathbf{x}_j + \boldsymbol{\Delta}_j)|^2 = |\mathbf{x}_i - \mathbf{x}_j|^2 + \underbrace{2(\mathbf{x}_i - \mathbf{x}_j) \cdot (\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j)}_{\text{first-order variation}} + |\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j|^2. \tag{2.27}$$

We are interested in displacements that have vanishing first-order variations, as these displacements do not introduce overlaps. For such displacements, in general $\boldsymbol{\Delta}_i \neq \boldsymbol{\Delta}_j$. A vanishing first-order variation indicates that the squared distance between disk i and j is increased by $|\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j|^2$, and the contact is lost. The first-order variation in eq. (2.27) can be written as a product of twice a sparse "escape matrix" $\mathcal{M}^{\text{esc}}$ of dimensions $N_{\text{contacts}} \times 2N$ with a $2N$-dimensional vector $\boldsymbol{\Delta} = \{\Delta_1^x, \Delta_1^y, \Delta_2^x, \Delta_2^y, \ldots\}$. Each row of $\mathcal{M}^{\text{esc}}$ corresponds to a contact. The row r corresponding to the contact between i and j has four non-zero entries

$$\begin{aligned} \mathcal{M}^{\text{esc}}_{r,2i-1} &= x_i - x_j, \\ \mathcal{M}^{\text{esc}}_{r,2i} &= y_i - y_j, \\ \mathcal{M}^{\text{esc}}_{r,2j-1} &= -(x_i - x_j), \\ \mathcal{M}^{\text{esc}}_{r,2j} &= -(y_i - y_j). \end{aligned} \tag{2.28}$$

The escape modes are found by solving

$$\mathcal{M}^{\text{esc}} \boldsymbol{\Delta} = 0 \tag{2.29}$$

using singular-value decomposition. Equation (2.29) allows factoring out the infinitesimal amplitude of the displacements $\boldsymbol{\Delta}$ and expressing the escape modes as the directions of the small displacements. For the $k = 5$ Böröczky packing with the Kahle core, we find 28 vanishing singular values. It follows from eq. (2.26) that, because of $28 = 2N - N_{\text{contact}}$, all contacts are linearly independent. The 28 modes are ranked by the cost function:

$$L = \sum_{i,j} (\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j)^2, \tag{2.30}$$

where the sum is over all contact pairs i and j. This function, acting on the $2N$ displacements $\boldsymbol{\Delta}$, measures the non-uniformity of a deformation. The resulting two lowest eigenmodes (with zero eigenvalue) of eq. (2.30) are due to the translational invariance in the periodic box. Other low-lying eigenmodes give smooth large-scale deformations which collectively escape the contact constraints (see Fig. 2.5). There are also escape modes that have positive first-order variations. Finding such modes amounts to solving a difficult linear programming problem. The escape modes point in the directions of going through the "wall" of locally stable packings in the sample space, suggesting that the locally stable packings do not block the sample space.

2.2.2.4 ε-relaxed Böröczky configurations

An ε-relaxed Böröczky configuration is obtained by shrinking the disks in a Böröczky packing by a factor of $1-\epsilon$. Fig. 2.6 shows an example of a ε-relaxed Böröczky configuration. One can also consider randomly displaced within a circle neighborhood of radius $\epsilon\sigma$. As we have seen in section 2.2.2.3, collective moves can escape the Böröczky packing. On the contrary, local Markov-chain Monte Carlo algorithms move one disk at a time, and may be completely blocked by the Böröczky packing. However, subsequent moves in the local algorithms can be arranged into collective moves. We formalize this arrangement by the ε-relaxed Böröczky configuration, as in practice individual moves accumulate only if there is space between the disks. The ε-relaxed Böröczky configurations can be used as a benchmark of the sampling algorithms and serve as a heuristic. It is discussed in the attached publication 2 and

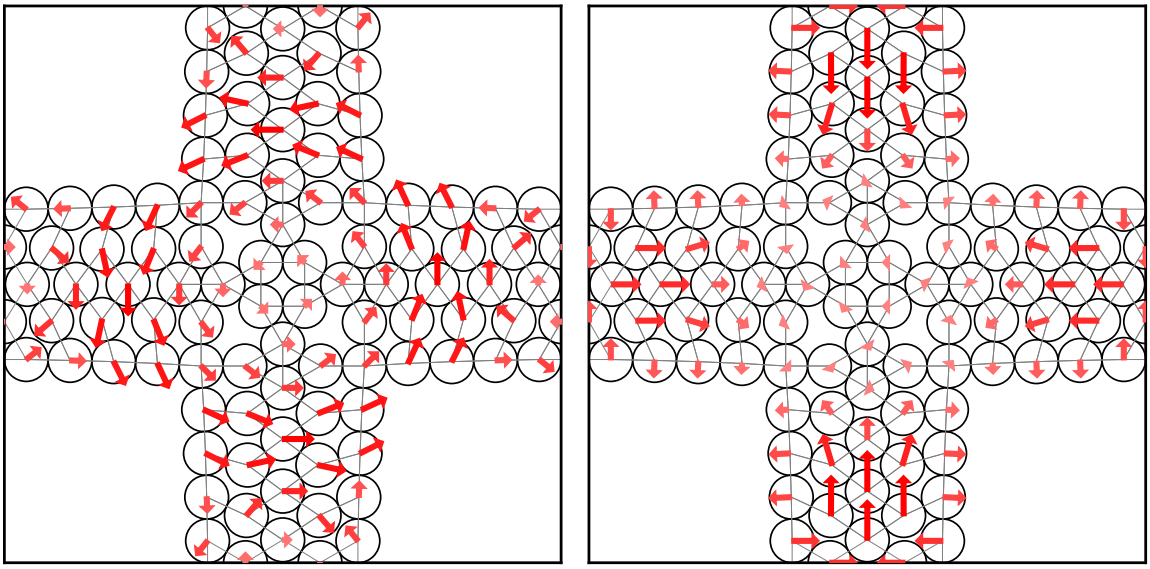

Fig. 2.5 *Two orthogonal modes (represented as red arrows) out of the 28-dimensional space of all collective escape modes $\boldsymbol{\Delta}$ for the $k = 5$ Böröczky packing with the Kahle core. Lines are drawn between pairs of disks which are in contact. (Figure from [46].)*

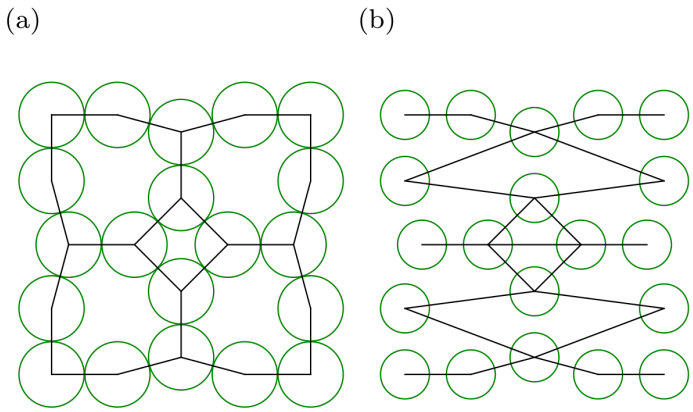

Fig. 2.6 (*a*) *Contact graph for an isolated Böröczky core.* (*b*) *Constraint graph* (*to be introduced in section 3.2.2*) *of an* ϵ*-relaxed Böröczky core with* $\epsilon = 0.25$. (*Figure from* [*46*].)

in chapter 3 that, in terms of escaping a ε-relaxed Böröczky configuration, the forward and Newtonian event-chain Monte Carlo algorithm outperform the straight variant qualitatively. The superior performance of the forward and Newtonian variant is recreated in more complicated systems [45].

Chapter 3

Algorithms

The Boltzmann distribution of hard disks in the sample spaces described in chapter 2 can be sampled by algorithms implementing Markov-chain Monte Carlo [5, 11, 60, 71, 84, 87] or, equivalently, from the Newtonian evolution implemented in molecular dynamics [1, 75, 104]. Although both classes of methods were introduced decades ago for the hard-disk model, the development of algorithms and of software implementing them has never stopped. In this chapter, we thus introduce and analyze the algorithms that sample the configurations in the hard-disk model, in particular, the event-chain Monte Carlo algorithms that we use and contribute to throughout this thesis.

We first discuss the aspects of the Markov chain related to computation, most importantly, the ergodic theorem that guarantees the convergence of the estimators introduced in chapter 4. Then we discuss the sampling algorithms, notably the event-chain Monte Carlo algorithms implementing a non-reversible Markov chain. On the one hand, we discuss our work on the parallelization of the software implementation of event-chain Monte Carlo variants which, for a multithreaded algorithm, is far from trivial. The full solution to the parallelization problem we provide in the attached publication 1 involves both an algorithmic aspect and an implementation aspect. In this chapter, only the algorithmic aspect is discussed. The algorithm is made practical by the techniques introduced in chapter 5. On the other hand, we discuss the dynamics of the Markov-chain Monte Carlo algorithm when escaping an ε-relaxed Böröczky configuration, introduced in chapter 2, and its scaling theory. This discussion follows closely the discussion in the section 3 in the attached publication 2 [46].

3.1 Markov-chain Monte Carlo: the basics

At low density $\eta \propto 1/N$, a configuration can be sampled by placing N disks in the box randomly and keeping the non-overlapping configuration. This procedure is referred to as direct sampling. These algorithms have recently been generalized for a small constant density in the $N \to \infty$ limit [121]. At density $\eta > 0.3$, the rejection rate of these approaches become too high, and practically no configuration can be sampled [62]. The remedy for this is to design algorithms modifying the existing configuration incrementally, referred to as Markov-chain Monte Carlo algorithms. The Markov-chain Monte Carlo algorithm generates samples from the stationary distribution [69]. For hard disks, the samples are the con-

figurations, and the stationary distribution corresponds to the uniform weight discussed in chapter 2. A correct sampling process requires the global-balance condition, irreducibility, and aperiodicity. If a Markov chain is irreducible, a unique stationary measure exists, which is the limit of the time-dependent measure if the Markov chain is also aperiodic.

Two convergences are important in the Markov-chain Monte Carlo. The first one is the convergence of distribution. At the beginning of the Markov chain, the distribution that a Markov chain samples is not the stationary distribution. The sampled distribution converges to the stationary distribution gradually, and this process is referred to as mixing. Mixing implies that the initial results produced by the sampling algorithms have to be thrown away. Usually, the result of computation is expressed as a numerical estimation of an interested quantity in physics. This quantity, denoted by O, is formulated as a function of the hard-disk configuration (more discussion on this topic in chapter 4). The second convergence is expressed in the form of the ergodic theorem for an irreducible Markov chain: the running average of O evaluated on the sampled configuration converges to the expectation of O. This theorem indicates that long enough Markov-chain Monte Carlo computations give unbiased results. Although the sample space for the hard-disk model contains an infinite number of configurations, only the Markov chain in a finite sample space is discussed in this subsection for simplicity. Also, the discussion is restricted to discrete-time Markov chains. As will be discussed in section 3.2.2, the continuous-time algorithm can be viewed as the limit of a discrete-time Markov chain.

3.1.1 Introduction to Markov chains

A Markov chain is a stochastic process in which the outcome of a step is only related to the outcome of the previous step. Denote the Markov chain as a sequence of random variables $X_0, X_1, ...$, for an arbitrary finite sample space Ω, the transition matrix is a matrix of conditional probabilities:

$$P(x, y) = P(X_{t+1} = y | X_t = x), \tag{3.1}$$

where $x, y \in \Omega$ are samples. $P(x, y)$ is the probability of choosing sample y when the outcome of the previous step is x. All of the entries of P are positive and

$$\sum_{y \in \Omega} P(x, y) = 1, \tag{3.2}$$

which is referred to as the stochasticity of matrix P. Since the Markov chain has no memory of the outcome before the previous step, the transition matrix, together with the initial condition, encodes all its information. Let the distribution of X_t be π^t, a vector labeled by all possible states in Ω, the distribution π^t is obtained by repeatedly applying transitions to the initial distribution π_0:

$$\pi^t = \pi_0 P^t. \tag{3.3}$$

An aperiodic and irreducible Markov chain can visit the whole sample space. Define

$$\mathcal{T}(x) = \{t \geq 1, P^t(x, x) > 0\}, \tag{3.4}$$

which is the least possible time of revisiting a state in the sample space. The period of the Markov chain is defined as the greatest common divisor of $\mathcal{T}(x), \forall x \in \Omega$. If the period of the

Markov chain is 1, then the Markov chain is aperiodic. For a Markov chain, if $\exists t$ such that $\forall x, y \in \Omega$, $P^t(x, y) > 0$, then the Markov chain is called irreducible. Irreducibility indicates that all states can be reached starting from an arbitrary state.

For each Markov chain, if the measure π satisfies

$$\pi = P\pi, \tag{3.5}$$

it is referred to as the stationary measure. As its name indicates, the stationary measure is invariant after transitions. The stationary measure is the Boltzmann measure introduced in chapter 2, and equation (3.5) is referred to as the global-balance condition.

Practically, all commonly used Monte Carlo algorithms satisfy the detailed-balance condition, a stronger condition than the global-balance condition. The detailed-balance condition is expressed as

$$\pi(x)P(x, y) = \pi(y)P(y, x). \tag{3.6}$$

Summing the expression of the detailed-balance condition over $x \in \Omega$ yields the global-balance condition:

$$\sum_{x\in\Omega} \pi(x)P(x, y) = \sum_{x\in\Omega} \pi(y)P(y, x) \tag{3.7}$$

$$\pi P = \pi. \tag{3.8}$$

The right-hand side of the second equality makes use of the stochasticity. Define the probability flow going from state x to state y as $\pi(x)P(x, y)$. The detailed-balance condition implies that the probability flow going from x to y is the same as the inverse probability flow going from y to x. Thus, when the direction of time is reversed, a Markov chain that satisfies the detailed-balance condition is still the same Markov chain, thus referred to as a reversible Markov chain. Checking the detailed-balance condition requires considering the transition between two states, while checking the global-balance condition requires examining all the states. Thus, verifying the detailed-balance condition is easier than verifying the global-balance condition. A Markov chain that breaks the detailed-balance condition is referred to as a non-reversible Markov chain. In simple models, non-reversible Markov chains accelerate sampling by turning the diffusive dynamics of reversible Markov chains into ballistic dynamics [10, 11, 86]. The event-chain Monte Carlo algorithms discussed in section 3.2.2 are non-reversible.

3.1.2 Convergence of Markov chains

The correctness of Markov-chain Monte Carlo algorithms is guaranteed essentially by two theorems: the convergence theorem and the ergodic theorem. The convergence theorem states that the distribution π^t converges exponentially to the stationary distribution if the Markov chain is irreducible and aperiodic. Converging to the stationary distribution is referred to as mixing, and the time it takes to mix a Markov chain is referred to as the mixing time. The ergodic theorem expresses that the running average of an observable converges to its expectation when the run is long enough.

3.1.2.1 Convergence theorem

The difference between two distributions π and π' can be quantified in many ways. One way commonly used in the mathematical literature is the total-variance distance, defined as

$$||\pi - \pi'||_{\text{TV}} = \frac{1}{2}\sum_{x\in\Omega}|\pi(x) - \pi'(x)|. \tag{3.9}$$

For a Markov chain that starts from a single state, for example x_0, the probability weights of the states at the time t is denoted as

$$\sum_{x\in\Omega}\delta(x - x_0)P^t(x, y) := P^t(x_0, \cdot). \tag{3.10}$$

The convergence theorem bounds the total-variance distance of the stationary distribution and the sampled distribution at time t [69]:

Theorem 1 *For an irreducible and aperiodic Markov chain P with stationary distribution π, there exist a constant $\alpha < 1$ and $C > 0$ such that*

$$\max_{x_0\in\Omega}||\pi - P^t(x_0, \cdot)||_{TV} \leq C\alpha^t. \tag{3.11}$$

The convergence theorem states that, for an irreducible and aperiodic Markov chain, even if starting from the worst initial configuration, the difference between π^t and the stationary distribution decays exponentially with respect to t. The mixing time is defined as

$$t_{\text{mix}} = \min\left\{t : \max_{x_0\in\Omega}||\pi - P^t(x_0, \cdot)||_{\text{TV}} \leq \epsilon\right\}, \tag{3.12}$$

where ϵ is an arbitrary positive number less than $1/2$. Configurations sampled before the mixing time are not guaranteed to be sampled from the stationary distribution. In the practical cases that we will discuss in section 3.3, the mixing time can be as long as ten years and is an important aspect regarding the performance of the algorithm.

The convergence theorem states that, for a long enough Markov chain, samples are practically identically distributed. However, it is not a statement regarding the independence of the samples. The configurations sampled from the uniform distribution have strong correlations in general, leading to difficulties in statistical analysis [114], which we will discuss in chapter 6.

3.1.2.2 Ergodic theorem

Let $O(x)$ be a real-valued function define in the sample space Ω. The ergodic theorem states that [69]

Theorem 2 *For an irreducible Markov chain, for any initial distribution π_0,*

$$P_{\pi_0}\left\{\lim_{T\to\infty}\sum_{t=0}^{T-1}O(X_t) = \sum_{x\in\Omega}\pi(x)O(x)\right\} = 1. \tag{3.13}$$

In chapter 4, we introduce $O(x)$ formally as an observable. The ergodic theorem states that, if the Markov-chain Monte Carlo algorithm runs long enough, the time average over the sampled configurations converges to the ensemble average. It implies that even though the hard-disk computation features randomness, the asymptotic behavior of the result is controlled. The ergodic theorem is a statement about a single run. It is not applicable to the average of runs. The ergodic theorem generalizes trivially to complex-valued functions.

In practice, the running average may not include sample at every step. For a Markov chain X'_t in the same sample space as X_t with transition matrix P' satisfying

$$P = P'^{\Delta t}, \tag{3.14}$$

where P is the transition matrix of the Markov chain X_t in (3.13), the sample obtained at each step for X_t is equivalent to the sample obtained every Δt steps for X'_t. The expression (3.13) can be written alternatively as

$$P_{\pi_0}\left\{\lim_{T\to\infty}\sum_{i=0}^{T-1} O(X'_{i\Delta t}) = \sum_{x\in\Omega}\pi(x)O(x)\right\} = 1. \tag{3.15}$$

Equation (3.15) implies that, if an observable is calculated every time interval Δt, its running average still converges to its expectation. We referred to the time at which the observable are calculated as $t = t_0, t_1, ..., t_{n_{\text{sample}}-1}$. This notation is particularly useful when discussing legacy methods of sampling a finite number of hard-disk configurations.

3.2 Sampling algorithms

A Markov-chain Monte Carlo algorithm generates a series of hard-disk configurations $\mathbf{x}(t)$, labeled by the Markov-chain steps $t = 0, 1, ..., T-1$ or continuous MCMC time introduced in section 3.2.2. Any Markov-chain Monte Carlo algorithm simply has to satisfy the global-balance condition of (3.5) and be irreducible. This implies that there is normally an enormous degrees of freedom for the choice of the correct transition matrices, of which, naturally, we can discuss only a few. The transition matrix not only governs the evolution of probability measures, but it also represents the algorithm which specifies how one sample evolves into the next one. In this section, we introduce to the sampling algorithms used during this thesis, in particular, the multithreaded event-chain Monte Carlo algorithm we have developed and presented in the attached publication 1. We also discuss the performance of local sequential Markov-chain Monte Carlo algorithms for hard disks, and propose their scaling theory, presented in section 3 in the attached publication 2 [46].

3.2.1 Metropolis algorithm and its massive parallelization

The earliest Markov-chain Monte Carlo algorithm is the Metropolis algorithm [84]. At each iteration of the algorithm, a random disk i is sampled from a uniform distribution among $1, ..., N$. Then, a move $\delta\mathbf{x}_i$ is proposed. The probability distribution $P_\delta(\delta x, \delta y)$ of $\delta\mathbf{x}_i$ satisfies

$$P_\delta(\delta x, \delta y) = P_\delta(-\delta x, -\delta y). \tag{3.16}$$

If the proposed move does not introduce any overlap with other disks or the wall (if any), the new configuration is kept, otherwise, the move is rejected, and the old configuration

at the beginning of the iteration is sampled again. In general, there is no constraint on the probability distribution of the proposed move other than (3.16). However, the proposed move in practice is usually small so that the rejection rate is kept low. In practice, the displacement $\delta\mathbf{x}_i$ can be, for example, a random vector aligned with $\hat{\mathbf{e}}_x$ or $\hat{\mathbf{e}}_y$ (cross move set) or in a square centered at the origin (square move set).

Running the Metropolis algorithm in parallel amounts to moving multiple disks at the same time. It is possible that independent moves of disks do not introduce any overlap, while two of the moved disks overlap with each other. Such overlaps are avoided by constraining the position of the moved disk. The resulting algorithm is the Massively Parallel Monte Carlo algorithm (MPMC), running on graphical processing units (GPUs), first proposed by Engel et al. [5]. It uses a four-color checkerboard of rectangular cells whose sides are larger than 2σ, aligned with the x and y axes, compatible with the periodic boundary conditions.[1] The distance between two disks in different cells of the same color is always larger than 2σ, and moving two disks in different cells simultaneously never introduces overlap between them. The MPMC algorithm samples one of the four colors, and then independently updates disks in all corresponding cells using the Metropolis algorithm with the additional constraint that disks cannot leave their cells (see Fig. 3.1). After a certain time, the color is resampled, and the checkerboard is reinitialized, which is necessary for the algorithm to be irreducible. Each cell can be viewed as a non-periodic box. MPMC satisfies the detailed-balance condition, as it is based on the Metropolis algorithm.

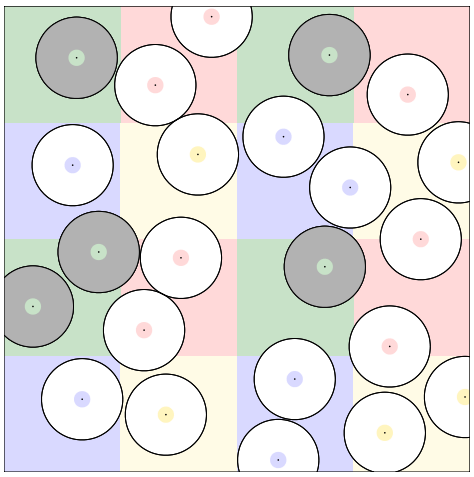

Fig. 3.1 *Four-color checkerboard decomposition in a periodic box, with cells larger than* 2σ*. If the* green *color is chosen, highlighted disks may move, but cannot leave their cells. Disks in different green cells do not communicate. The checkerboard allows massive parallelization.* (*Figure from* [70].)

3.2.2 Event-chain Monte Carlo algorithms

The event-chain Monte Carlo (ECMC) algorithm is proposed in [11]. It is a continuous-time event-driven sampling algorithm described by an non-reversible lifted Markov chain.

[1] The MPMC usually treats the large-N case in a periodic box. We do not consider small N and non-periodic systems for MPMC.

Except for parallelized ECMC, only one disk moves at a time. The moving disk is referred to as "active". It moves in a straight line, while the other disks are at rest and referred to as "static". The index of the active disk and its velocity are updated at collisions. At each transition in terms of the Markov chain, the active disk moves by an infinitesimal amount. The discrete Markov-chain steps are mapped onto continuous MCMC time, with two steps separated by an infinitesimal time interval. For a finite period of MCMC time, the active disk moves a finite distance. The number of sampled configurations is infinite and proportional to the interval of MCMC time. The infinite number of configurations are only conceptual, and only configurations at collisions are computed.[2] In each step, the system changes from a configuration at collision to the configuration at the next collision. The ECMC algorithm samples the lifted configurations space [63], which is a product of the sample space of hard-disk configurations Ω, and the space of lifting variables. The latter are usually the index and the velocity of the active disk. The detailed-balance condition is broken in ECMC, and there are no rejections. Depending on the choice of lifting variables and the rule of updating the velocities in collision, there are multiple variants of ECMC, namely straight, reflective [11], forward [87], and Newtonian [60]. The moves in ECMC algorithms are deterministic, while the lifting variables are resampled after a MCMC time interval. The time interval between two resamplings is referred to as chain time (or chain length). It is shown that the randomness at resampling is essential for the ECMC algorithms [45]. The Böröczky packings block ECMC moves. However, as discussed in section 4.2.1 in the attached publication 2 [46], the ECMC algorithms can still be used in practice.

3.2.2.1 Straight ECMC

Straight ECMC [11] is one of the first proposed ECMC algorithms, and it is one of the primary methods for hard-disk pressure computations [10, 29, 70]. The lifted configuration in straight ECMC is $(\mathbf{x}, a, \mathbf{d})$, where a denotes the index of the active disk, and $\mathbf{d}$ is a unit vector denoting the direction of the move, conventionally taking values from $\{\hat{\mathbf{e}}_x, \hat{\mathbf{e}}_y\}$ for a periodic box and $\{\pm\hat{\mathbf{e}}_x, \pm\hat{\mathbf{e}}_y\}$ for a non-periodic box. As the velocity is of unit norm, the MCMC time in the straight ECMC is equivalent to the distance covered by the active disk. The active disk moves along the direction $\mathbf{d}$ in a straight line until the collision takes place. For a collision between the active disk $a = i$ and another disk j, the velocity remains the same, and the active disk becomes j after the collision. In the non-periodic box, when the active disk collides with the wall, its direction $\mathbf{d}$ is updated such that the component perpendicular to the wall changes sign.

The straight ECMC satisfies the global-balance condition. For systems with low density, the active disk may not be able to find any collision partner, and it can go through the periodic box repeatedly, resulting in periodicity that is practically resolved by resampling. At the phase-transition density, such periodic behaviors are not observed. Configurations with strong symmetries, for example perfectly aligned disks, can also invoke periodicity even at high density. In practice, this can be avoided by randomly displacing the perfectly aligned disks by a tiny amount. It can also be avoided by making the chain length random. The active disk moves in both x and y direction so that both the x and y components of the disk positions are updated to achieve irreducibility.

[2] As discussed in chapter 4, these conceptual configurations contribute to the computation of pressure. We label these conceptual configurations by Markov-chain steps $t = 0, 1, ..., T-1$.

Straight ECMC is prone to gridlock. In a gridlock, the active-disk label loops through a subset of N disks being in contact. The active disk get updated all the time, but there is no displacement. As a consequence, the MCMC time remains the same during the gridlock. As the resampling of the active disk and velocity takes place at specific MCMC time, it will never take place to resolve the gridlock. The gridlock may happen in a number of configurations in straight ECMC, for example, for any configuration that has a ring of contacting disks winding around the periodic box. Gridlocks are also observed for straight ECMC from tightly packed initial configurations [120, Section 4.2.3]. We have further observed that the gridlock occasionally appears when the straight ECMC starts from a ε-relaxed Böröczky configuration in our arbitrarily precise implementation (see Appendix A.3 in the attached publication 2 [46]). However, according to our experience, gridlock does not appear in the practical usage of the straight ECMC.

The straight ECMC can be formulated in terms of a constraint graph [57, 71], a Verlet list [117] that remains unchanged forever if the active disk moves in only $\hat{\mathbf{e}}_x$ or $\hat{\mathbf{e}}_y$. For a given initial condition and velocity $\mathbf{d} = \hat{\mathbf{e}}_x$, arrows $[i \to j]$ of the constraint graph $\mathcal{G}_{\hat{\mathbf{e}}_x}$ represent possible collision between disk i and disk j [57]. When scheduling collisions, only the disks pointed by the arrows are considered. Each disk has at most three arrows (see an example in Fig. 3.2(a)). These arrows are found by identifying the nearest disk in the x direction for the disks in three lanes, defined as intervals of the relative position in the y direction. In the resulting constraint graph, referred to as $\mathcal{G}^{(3)}$, every disk has three disks that it can potentially collide with. As the three potential collision partners may block each other, the number of arrows corresponding to potential collisions is less than three. As demonstrated in Fig. 3.2(a), there are even non-local arrows. According to experiment, the number of genuine collision arrows for each disk is on average approximately 2.1 at a density around $\eta = 0.7$ (see Fig. 3.2(c)). The redundant arrows can be removed (see Fig. 3.2(b)), but in most cases the performance gain by having less arrow is smaller than the performance loss by pruning the redundant arrows. In practice, the straight ECMC requires the direction of the active disk to be changed after a rather small number of collisions [57]. Thus, the constraint graphs have to be recomputed frequently. A cell system can be implemented in place of the constraint graph.

The straight ECMC can be parallelized, thanks to the restrictive direction of displacements and the small number of potential collision partners. The parallel version of straight ECMC is presented in section 3.2.3

3.2.2.2 Reflective ECMC

The reflective ECMC [11] was proposed together with straight ECMC. The lifted variables are an index of the active disk and a unit vector of arbitrary direction, serving as the velocity of the active disk. At the collision, the velocity is reflected with respect to the line connecting the center of the two involved disks. Let the velocity of the active disk be $\mathbf{v}$ and the jth disk be hit by the active disk i. Denote the vector connecting two disks at collision as $\mathbf{r}_{ij} = \mathbf{x}_j - \mathbf{x}_i$. The new velocity $\mathbf{v}'$ after the collision is

$$\mathbf{v}' = -\mathbf{v} + \frac{\mathbf{r}_{ij}}{2\sigma^2}\mathbf{r}_{ij} \cdot \mathbf{v}. \tag{3.17}$$

The reflective ECMC is proven to satisfy the global-balance condition. Resampling is needed after every fixed period of MCMC time. Similar to the straight variant, the reflective ECMC

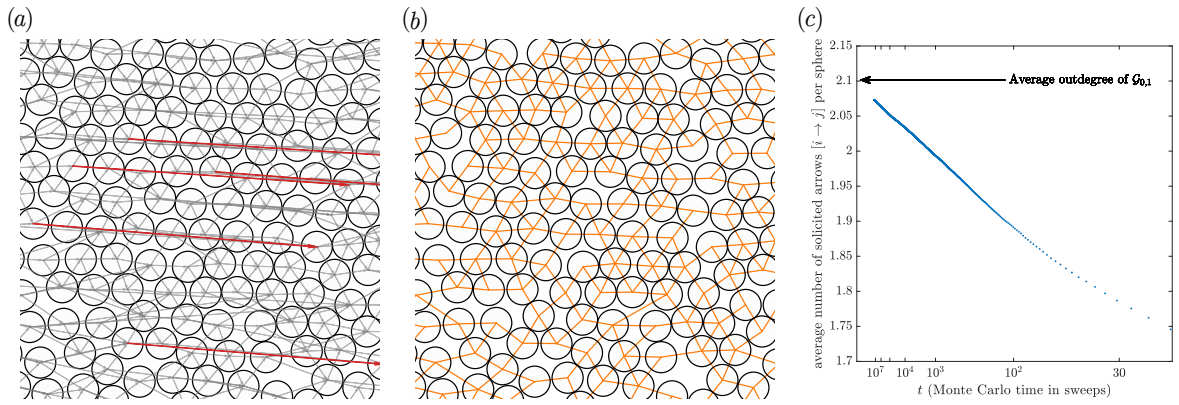

Fig. 3.2 *ECMC constraint graphs for a $N = 256^2$ system. (a): $\mathcal{G}^{(3)}$ for this configuration (detail), with highlighted non-local arrows. (b): Constraint graph after pruning. (c): Number of solicited arrows found in a long cell-based ECMC run. The constraint graph serves not only as a neighbor list but also as the key to enforcing causality in multithreaded ECMC. (Figure from [71].)*

is vulnerable to gridlock.

3.2.2.3 Forward ECMC

The forward ECMC [87] has the same lifting variables as the reflective ECMC. The only difference between the forward ECMC and reflective ECMC is how the velocity is updated at the disk collisions. The component orthogonal to the line connecting the disks at contact is uniformly sampled between 0 and 1, and its sign is chosen to preserve the overall direction of the velocity before the collision. Its parallel component is determined such that the velocity is of unit norm. The sign of the parallel component remains unchanged. Let u be a random number sampled from a uniform distribution in $(0, 1)$. Denoting $\mathbf{n}$ as the unit vector perpendicular to the line connecting two disks at collision, namely $\frac{1}{2\sigma}(r_{ij,y}, -r_{ij,x})$, using the same notations as in the discussion of the reflective ECMC, the velocity after collision $\mathbf{v}'$ is calculated such that

$$\mathbf{v}' = \mathrm{sgn}(\mathbf{r} \cdot \mathbf{v})\sqrt{1 - u^2}\frac{\mathbf{r}_{ij}}{2\sigma} - \mathrm{sgn}(\mathbf{n} \cdot \mathbf{v})u\mathbf{n} \tag{3.18}$$

The forward ECMC is proven to satisfy the global-balance condition. There is no proof that the resampling can be waived for the forward ECMC, despite that it is the only ECMC algorithm possessing per-event randomness. The per-event randomness prevents gridlocks in forward ECMC.

3.2.2.4 Newtonian ECMC

In Newtonian ECMC [60], each disk has a velocity which is, however, for the static disks only a label, and does not always correspond to a change of position with time. There is only one active disk, moving according to its own velocity. At a collision, the velocities of two colliding disks are updated as in Newtonian dynamics, discussed in chapter 2. As there is only one disk moving, the slow disk is allowed to collide with a faster disk moving in the

same direction, leading to impossible collisions in realistic Newtonian dynamics. Resampling all of the velocities and the active disk is required in fixed intervals of MCMC time. Besides that, it suffers from frequent gridlocks when the initial configuration is a ε-relaxed Böröczky configuration with small ϵ.

3.2.3 Multithreaded ECMC

In the attached publication 1, we have parallelized the straight ECMC, resulting in the multithreaded ECMC algorithm. The first step of parallelizing the straight ECMC is to allow multiple active disks. In multithreaded ECMC, each lifted configuration contains the hard-disk configuration, the direction of all the active disks, and a set A of cardinality N_A containing indices of the active disk. In the continuous-time description of multithreaded ECMC, all of the active disks move with the same velocity, and there is no collision between two active disks. This is why we choose to begin the parallelization of ECMC algorithms from the straight variant. When there is a collision between an active disk and a static disk, the velocity is updated as in the straight ECMC. Similar to straight ECMC, the MCMC time is the distance covered by each of the active disks, which is referred to as the global time. The continuous-time process is referred to as the Continuous Process with Global Time in the attached publication 1 [71], and shown in Fig. 3.3(a). We have proven that this process satisfies the global-balance condition.

Each of the active disks is associated with a thread, and the computation on each thread is independent.[3] All the threads access the same system, in the sense that a thread can "see" the displacements on all other threads. The independent moves on multiple threads should process exactly the same collisions as in the Continuous Process with Global Time. This requires the disks to be moved according to a specific order. In straight ECMC, the ECMC time is equivalent to the distance covered by the active disk. However, in the multithreaded ECMC, the active disks move independently, and there is no longer a well-defined time in the system and across all threads. Each active disk has its own time. The position of active disks are asynchronized, and an active disk being advanced in time can hit a static disk which is supposed to collide with another active disk, breaking the order specified by the continuous process with global time, and creating issues regarding causality. To solve this problem, we assign a time to each of the disks, active or not, as the local time—the MCMC time at the last moment when it is active. Let the disk i be an active disk, j be a potential collision partner of disk i, and τ_{ij} be the MCMC time it takes for disk i to hit disk j. The horizon condition is defined as

$$t_i + \tau_{ij} > t_j, \tag{3.19}$$

where t_i and t_j are the local times of disk i and j respectively. If the horizon condition is violated by any potential collision partners (specified in the constraint graph), this collision breaks causality. If the causality is preserved throughout the run, the outcome of the multithreaded process is identical to the outcome of the continuous process with global time. The multithreaded ECMC is thus described as the following process: all of the active disks move independently. The velocity and active disk are updated as in straight ECMC at each collision. The horizon condition is checked before each collision. If there is no violation of

[3] There can be multiple active disks on a thread. However, for simplicity, we always assume a thread is in charge of a single active disk in the discussion of algorithms.

the horizon condition until the end of the run, all the output during the run is kept. Otherwise, no output is recorded, and the run restart from the initial configuration. This process is shown in Fig. 3.3(b). To keep the horizon-violation rate low, each chain in the multithreaded ECMC is further broken by breakpoints. The time between two breakpoints is short, and the multithreaded ECMC evolves the system from a breakpoint to the next one repeatedly. The detailed discussion of the multithreaded ECMC can be found in section 2 in the attached publication 1 [71].

The continuous process with global time satisfies the global-balance condition. Thus, the multithreaded ECMC algorithm with local time also satisfies the global balance condition. The practical irreducibility and aperiodicity (when locally sparse packings are ignored) of the multithreaded ECMC is inherited from the straight ECMC. In terms of parallelization, we prefer the straight ECMC over the other variants mainly for two reasons. The first is that the active disk never collides in the continuous process with global time. In straight ECMC, the active disk moves in a single direction with the same velocity, while in other variants, the active disk moves in arbitrary directions. We believe that there can be velocity-update schemes that respect the global balance and allow collisions between active disks. However, such a scheme has not been implemented. The second reason is the existence of the constraint graph in the straight ECMC. The horizon condition requires comparing the local time of the active and its possible neighbor. In straight ECMC, there are at most three neighbors to be checked at each collision. However, the other variants require scanning the whole system, leading to a $O(N)$ complexity for each event at least if implemented naively.

3.2.4 Molecular Dynamics

The most intuitive sampling algorithm is to reproduce the hard-disk Newtonian dynamics. This process is referred to as molecular dynamics (MD) [1]. For the hard-disk model, MD can be implemented event-driven. The positions of the disks at the next collision, either between the disks or between a disk and a wall, is computed. The velocities are updated at collisions by the law of Newtonian dynamics. In practice, MD can be used for either simulating realistic hard-disk trajectories or generating hard-disk configurations.

EDMD is, in its nature, asynchronized, rendering parallelization non-trivial. There is currently no valid parallel implementation of EDMD, though a number of parallelization schemes were proposed [76]. Although the multithreaded ECMC and the EDMD are similar in nature, the parallelization of ECMC is successful, because the number of moving disks is of $O(1)$, significantly smaller than N in EDMD.

3.3 Algorithm performance

In this section, we discuss the algorithm performance, measured by the number of iteration in an algorithm to finish a specific task, of Markov-chain Monte Carlo algorithms. We are interested in two tasks: escaping from an ε-relaxed Böröczky configuration and decoupling from the initial (dis)order in a large hard-disk system. The former task highlights the benefit of having a larger move set and no intrinsic scale in the reflective, forward, and Newtonian variants of ECMC, while the latter task demonstrates the overwhelming speedup introduced by the straight ECMC compared to the Metropolis algorithm. All the Markov-chain Monte

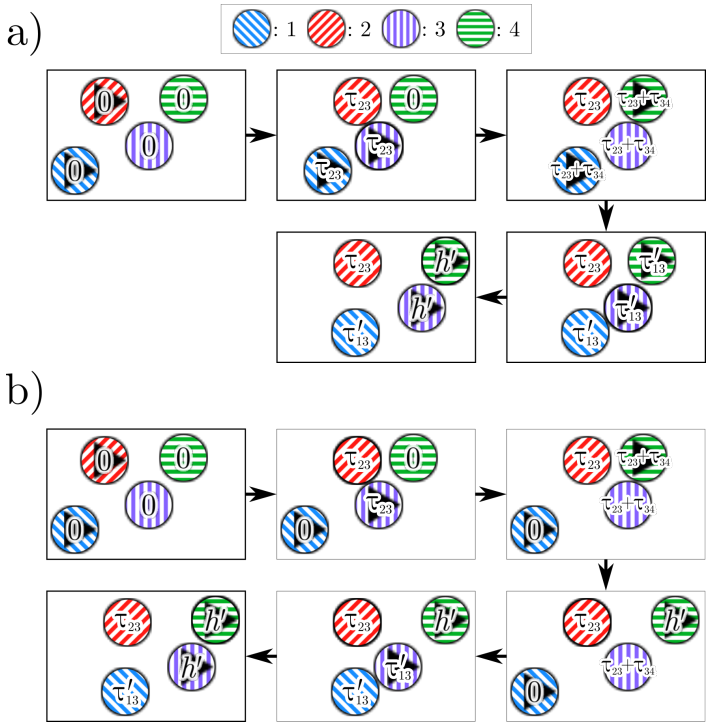

Fig. 3.3 *Examples trajectories related to multithreaded ECMC in a system containing four disks, of which two are active. The black triangle on the disks implies that the disk is active. The number on each disk is its local time.* (*a*) *A demonstrative trajectory of the continuous process with local time, presented in an event-driven way. Although the MCMC time is well- defined for this process, the times on each disk are their local time.* (*b*) *A demonstrative trajectory of the multithreaded ECMC. There is no violation of horizon condition, and the final configurations of* (*a*) *and* (*b*) *are the same.* (*Figure from* [71].)

Carlo algorithms that we discuss here differ only in some details, but their performance can be drastically different.

3.3.1 Escaping performance

The Böröczky packing blocks the local Markov-chain Monte Carlo algorithms completely, and it is also difficult to break the Böröczky packing structure in ε-relaxed Böröczky configurations. Clearly, if a sampling algorithm is trapped in a Böröczky packing, the distribution of sampled configurations is different from the stationary distribution. The iterations it takes to escape an ε-relaxed Böröczky configuration thus provides a lower bound of the mixing time.[4] We run the sampling algorithms using an ε-relaxed Böröczky configuration as the initial configuration.

Define the maximum nearest-neighbor distance

$$d(t) = \max_i \left[\min_{j(\neq i)} |\mathbf{x}_{ij}(t)| \right], \tag{3.20}$$

where $|\mathbf{x}_{ij}(t)| = |\mathbf{x}_j(t) - \mathbf{x}_i(t)|$ is the distance between disks i and j (possibly corrected for periodic boundary conditions). The maximum nearest-neighbor distance implies how far the loosest disk is away from the packing. When the maximum nearest-neighbor distance is large enough, at least one disk has broken free. The packing will be completely broken consequently. The t in (3.20) is the number of iterations, i.e. the number of proposed moves in the Metropolis algorithm and the number of events in the ECMC algorithms. We define the escape time t_{esc}, an integer, as the time t at which $d(t)$ has increased by ten percent:

$$t_{\text{esc}} = t : d(t) > 2.2\sigma, \tag{3.21}$$

More detailed discussion of escape time can be found in section 3.2.1 in the attached publication 2 [46].

As the distances between disks are small in the ε-relaxed Böröczky configuration, the required total displacement for escaping an ε-relaxed Böröczky configuration is small. In practice, the Newtonian ECMC, reflective ECMC, and forward ECMC do not require frequent resampling, and their chain length can be viewed as infinite compared to the total displacement. As the number of constraints in the Böröczky packing is larger than N, the single-direction moves in straight ECMC are not efficient for the escape from ε-relaxed Böröczky configurations. The direction in straight ECMC has to change frequently, introducing an intrinsic length τ_{chain}, the MCMC time of each chain, into the algorithm. Also, the Metropolis algorithm has to propose small moves to keep the rejection rate low. The length scale δ of the proposed move is the intrinsic length scale in the Metropolis algorithm. We numerically measure the escape time t_{esc} of these algorithms for a wide range of their intrinsic parameters and for small relaxation parameters ε (see Fig. 3.4, for the escape times from ε-relaxed Böröczky configurations). The escape time is a "V"-shaped function with respect to the intrinsic scale. We have proposed a scaling theory for this. To escape ε-relaxed Böröczky configurations, the accumulated displacement has to be large enough, which favors large δ or τ_{chain}. In the meantime, only moves on the scales ϵ are helpful, which favors small δ or

[4] In section 3.1.2.2, the mixing time is defined in Markov-chain steps. However, it is convenient to measure the mixing time in the number of events in an event-driven algorithm or in computer time during benchmarks.

τ_{chain}. The "V"-shaped function is a result of the competition of the two requirements. The scaling of optimal intrinsic scale is achieved when the two requirement are equally strong. The detailed discussion of the intrinsic scale can be found in section 3.2.2 in the attached publication 2 [46]. When inserting the scaling of the optimal intrinsic scale, the scaling of escape time is

$$t_{\text{esc}} \sim \begin{cases} \varepsilon^{-1} & \text{(Metropolis—square)}, \\ \varepsilon^{-2/3} & \text{(Metropolis—cross)}, \\ \varepsilon^{-2/3} & \text{(straight ECMC)}, \end{cases} \qquad \text{(for optimal } \delta^{\text{min}}, \tau^{\text{min}}_{\text{chain}}\text{)}. \tag{3.22}$$

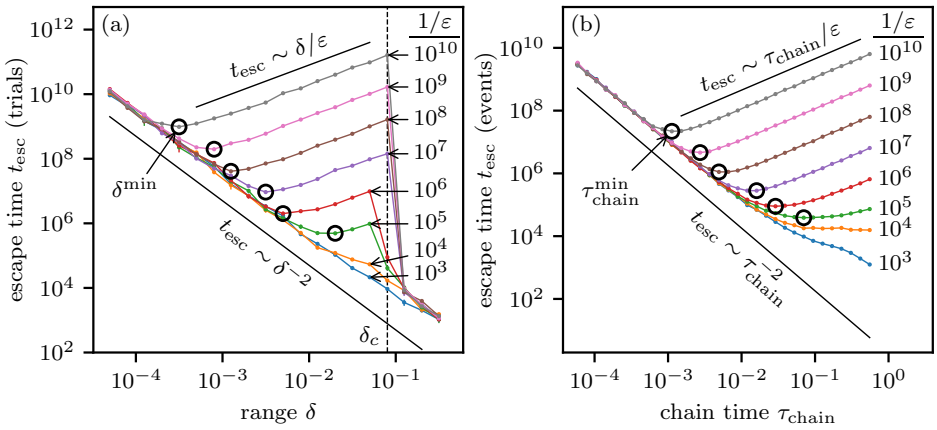

Fig. 3.4 *Median escape times from the $k = 5$ ε-relaxed Böröczky configuration ($N = 96$ disks) for different ε. (a) t_{esc} (in proposed moves) vs. range δ for the Metropolis algorithm with the cross-shaped displacement set. (b) t_{esc} (in events) vs. chain time τ_{chain} for straight ECMC. Error bars are smaller than the marker sizes (Figure from [46].)*

The forward, reflective, and Newtonian variants of ECMC have no intrinsic scale when escaping an ε-relaxed Böröczky configuration. Without an intrinsic parameter, the effective free path between events may thus be adaptive. The free path is initially on the scale ε, but then grows on average by a constant factor at each event, reaching a scale $\varepsilon' > \varepsilon$ after a time (that is, after a number of events) that scales as $\sim \ln(\varepsilon'/\varepsilon)$. The scale ε' at which the algorithms break free is independent of the initial scale ε. The theory of the effective free path is confirmed by computation. The detailed discussion on the effective free path can be found in section 3.2.3 in the attached publication 2 [46]. The theory of effective free path predicts a logarithmic scaling of the escape time (measured in events):

$$t_{\text{esc}} \sim \ln(1/\varepsilon) \quad \text{(reflective, forward, and Newtonian ECMC)}. \tag{3.23}$$

The numerical scaling of escape time is shown in Fig. 3.5. The scaling of the escape time indicates that, for specific initial configurations, the reflective, forward, and Newtonian ECMC have a large advantage compared to the straight ECMC, leading us to conjecture that the local Markov-chain Monte Carlo algorithms can have vastly different behaviors. The straight ECMC benefits from quick computation for each event and parallelization, which are the consequence of its unidirectional move. However, the scaling of escape time shows that the unidirectional move and intrinsic scale can be harmful in specific circumstances. Thus, the comparison of performance among all ECMC variants has no ultimate verdict.

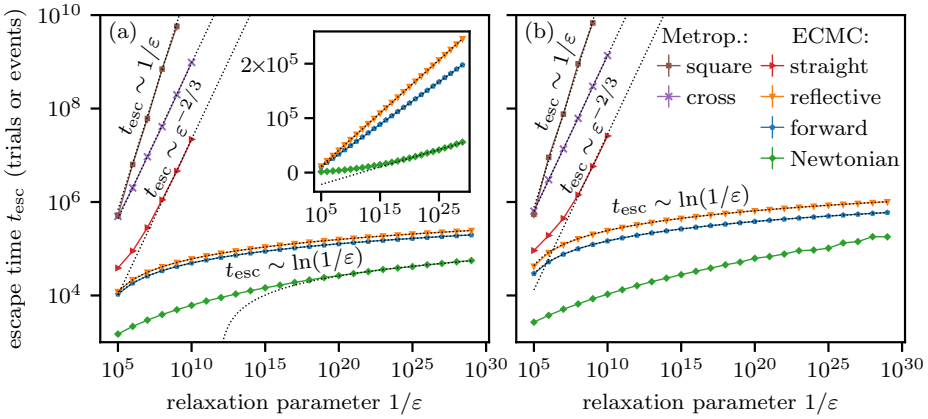

Fig. 3.5 *Median escape time t_{esc} from $k = 5$ ε-relaxed Böröczky configurations with different cores for local MCMC algorithms (where applicable: with optimized intrinsic parameters). (a) t_{esc} (Kahle core, $N = 96$ disks) for the Metropolis algorithm and straight ECMC. Inset: log–lin plots suggesting logarithmic scaling for the forward, reflective, and Newtonian ECMC. (b) t_{esc} for the Böröczky core ($N = 112$ disks). Error bars are smaller than the marker sizes. (Figure from [46].)*

3.3.2 Coarsening performance

For large N and at density $\eta \sim 0.708$, the hard-disk model at equilibrium is at coexistence of the liquid and hexatic phases. However, generating a configuration at coexistence is hard because of the phenomenon of coarsening. The initial configuration is usually a perfect crystal, or a disordered configuration obtained by the simulated annealing algorithm [77] for the hard disk model. It is thus interesting to see how long does it takes to reach the mixture from either an ordered phase or an disordered phase. The phase of the system is described by its global orientational order Ψ_6 that will be introduced in detail in section 4.4. For a perfect crystal $|\Psi_6| = 1$, and for a disordered configuration $|\Psi_6| = 0$.

Also, the coarsening provides a bound for the mixing time. At density $\eta = 0.708$, the typical configuration has $|\Psi_6| \sim 0.45$. If the value of $|\Psi_6|$ during a run stays at roughly 0 or 1, the Markov chain is not mixed, as the distribution is biased significantly towards disordered or ordered configurations.

The Ψ_6 as a function of number of sweeps for density $\eta = 0.708$ and $\eta = 0.718$ is shown in Fig. 3.6. For each setup, there is a run starting from the crystal initial configuration and a run starting from a disordered configuration. Comparing the result given by Metropolis algorithms (or MPMC) and by straight ECMC, one concludes that reaching the target $|\Psi_6|$ takes roughly 10^3 more sweeps for the Metropolis algorithm and MPMC than the straight ECMC for both densities. Therefore, in the range of density of interest, in terms of the number of proposed moves (events), the straight ECMC is 10^3 faster than the Metropolis algorithm, although their implementation only differ in some apparently minor details.

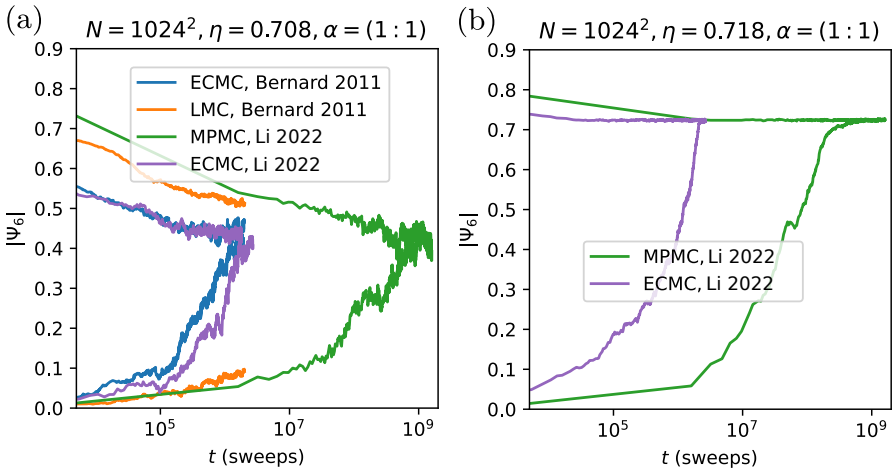

Fig. 3.6 *The $|\Psi_6|$ as a function of number of sweeps for density (a) $\eta = 0.708$ and (b) $\eta = 0.718$, starting with an crystal or an disordered configuration. The data is obtained using MPMC and straight ECMC. In terms of number of sweeps, the ECMC is 10^3 times faster than the MPMC. (Figure from [70].)*

Chapter 4

Observables

Understanding the hard-disk model requires more than just obtaining the samples $\mathbf{x}$ in Ω, namely the calculation of functions of the samples (what in mathematics are called "statistics"). The analysis of the model also requires computing quantities such as the distribution of hard disks, the pressure, and the orientational order. These quantities are referred to as observables which themselves have probability distributions. In the large-N limit, these distributions become sharply peaked. Examples for functions of the data are distribution functions, free energies, pressures, correlation functions, and are characterized by their expectation values. We are thus naturally interested in the estimators of expectation values even for finite systems, and will simply call them "estimators". In this chapter, we introduce the definition of observables and the corresponding estimators we have used in this thesis.

This chapter starts with the definition of observables, followed by discussions regarding specific observables. In particular, we introduce to the position distributions of the hard disks, whose computation is described in detail in Appendix A.1 in the attached publication 3 [70]. We then discuss the two definitions of pressure and derive pressure estimators, that is, estimators of their expectation value verified by computations with unprecedented precision in the attached publication 3. These observables avoid extrapolation and are not sensitive to the choice of statistical ensemble. We also discuss the orientational order of hard disks, an order parameter of the hard-disk model which correlates to pressure in a finite system.

4.1 Overview of observables

Here we introduce the general ideas and notations regarding the observable and corresponding estimator. Also, we show that the ergodic theorem of the Markov chain guarantees that the estimator of an observable converges to its expectation value. An observable is a complex-valued function of hard-disk configurations defined in the sample space

$$O : \Omega \to \mathbb{C}. \tag{4.1}$$

The expectation of an observable is

$$\langle O \rangle = \int_{\Omega} O(\mathbf{x}) \pi(\mathbf{x}) d\mathbf{x}, \tag{4.2}$$

where $\pi(\mathbf{x})$ is the probability density function of configuration $\mathbf{x}$. The probability distribution of O depends on parameters of the model, and so does its expectation. Given a sequence of samples $\{\mathbf{x}(t), t = 0, 1, ..., T-1\}$[1], the estimator of the expectation is the observable evaluated and averaged in this sequence, namely

$$\hat{O} = \frac{1}{T} \sum_{\mathbf{x}' \in \{\mathbf{x}(t), t=0,1,...,T-1\}} O(\mathbf{x}') = \frac{1}{T} \sum_{t=0}^{T-1} O(\mathbf{x}(t)). \tag{4.3}$$

The ECMC algorithms and molecular dynamics are continuous methods and sample uncountable numbers of configurations. (The sampled configurations are as much as the real numbers.) In these algorithms, the estimator can be written in the forms not compatible to (4.3). However, the derivation of these estimators still originates from (4.3) and requires discretizing the continuous process. The probability density function in (4.2) disappears in (4.3), since the sequence $\{\mathbf{x}(t), t = 0, 1, ..., T-1\}$ is sampled from the stationary distribution $\pi(\mathbf{x})$ of the Markov chain. By the ergodic theorem of Markov chains, presented in section 3.1.2, the running average of an observable O converges to its expectation $\langle O \rangle$. Although the expectation is not the only aspect of an observable, it is the most important one. Interesting properties such as equation of state and orientational order can both be formulated as expectations. Also, in the thermodynamic limit $N \to \infty$, the probability measure of an observable concentrates around its expectation value, which in turn carries all information. Besides, we are not only interested in the estimation of expectation values. As will be discussed in section 4.4, the expectation of the global orientational order is known. In this case, the sequence $\{O(\mathbf{x}(t)), t = 0, 1, ..., T-1\}$ be used as a diagnostic tool for ergodicity, as will be discussed in chapter 7.

4.2 Distributions

Distributions (of disk positions) are observables, and can either be represented by the probability density function or the cumulative distribution function. They are easy to evaluate compared to the pressure and the orientational order. They show the preferred position in the box, and are involved both in the computation of pressure (discussed in section 4.3) and in the validation of sampling algorithms (discussed in chapter 6). The position can refer to either the absolute position in the box or the relative position with respect to another disk. The distributions of the two possible choices of position are the position distribution and pair-correlation function, respectively. We will show that the histogram of the distributions of positions can be defined as an observable as formulated in section 4.1. The histogram, with the help of extrapolation, can be used to approximate the continuous probability density function. As will be discussed in section 4.3, the probability density can be computed without binning and extrapolation in the continuous sampling method such as ECMC. In the attached publication 3, we use the method involving binning only for cross validation. Nevertheless, the continuous-time methods are limiting cases of vanishing bin size and infinite sampled configurations. In this section, we introduce the finite bin-size case to provide a general picture.

[1]Sometimes, configurations are sampled after every sampled interval at $t = t_0, t_1, ..., t_{n_{\text{sample}}-1}$, instead of at each step in the Markov chain $t = 0, 1, ..., T-1$.

4.2.1 Position distribution

The position distribution refers to the two-dimensional probability density function of the position $\mathbf{x}_i = (x_i, y_i)$ of a single disk $i = 1, 2, ..., N$. As all disks are identical, the distributions of all disks are the same. Usually, the position distribution is used in a non-periodic box. In a periodic box, the position distribution is uniform due to the translational invariance.[2] The position distribution can be integrated in the x or y direction and yields the marginal position distribution in either y or x direction, respectively. The aspect ratio other than one leads to different marginal position distributions in the x and y direction. For simplicity, we set the direction to x for the unidirectional discussion. In practical computations, the position distribution is expressed by a histogram, and sometimes with the help of a curve fitting the histogram.

4.2.1.1 Overview of hard-disk position distribution

Here, we discuss the generals aspects of the position distribution, providing background for the discussion of pressure estimators in section 4.3 and sampling algorithm comparison in chapter 6. The position distribution is expressed by $\pi^{(1)}(\mathbf{x})$ (equivalently $\pi^{(1)}(x, y)$) such that the probability of having a single disk in a small region $[x, x + dx] \times [y, y + dy]$ is $\pi^{(1)}(x, y)dxdy$. $\pi^{(1)}(x, y)$ is a probability density function, meaning that it is positive everywhere in $[0, L_x] \times [0, L_y]$, and

$$\int_0^{L_x} \int_0^{L_y} \pi^{(1)}(x, y)dxdy = 1. \tag{4.4}$$

Function $\pi^{(1)}(x, y)$ is discretized by dividing the box into bins. Let n_b be the number of bins in both x and y direction. A mesh of spacing $\Delta x = L_x/n_b$ in the x direction and $\Delta y = L_y/n_b$ in the y direction split the box into n_b^2 bins. Denote the bin $[(i-1)\Delta x, i\Delta x] \times [(j-1)\Delta y, j\Delta y]$, $i, j = 1, 2, ..., n_b$, as the bin (i, j), the probability of having a disk in the bin (i, j) is

$$\text{Prob}_{ij} = \int_{(i-1)\Delta x}^{i\Delta x} \int_{(j-1)\Delta y}^{j\Delta y} \pi^{(1)}(x, y)dxdy. \tag{4.5}$$

The corresponding observable is

$$p_{ij}(\mathbf{x}) = \sum_{k=1}^{N} \Theta_{ij}(\mathbf{x}_k)/N, \tag{4.6}$$

where the bin member function

$$\Theta_{ij}(\mathbf{x}_k) = \begin{cases} 1, & (i-1)\Delta x < x_k < i\Delta x \text{ and } (j-1)\Delta y < y_k < j\Delta y; \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

It is readily checked that the expectation of p_{ij} is Prob_{ij}. For n sampled configurations, the estimator for the observable p_{ij} is

$$\hat{p}_{ij} = \frac{n_{ij}}{N n_{\text{sample}}}, \tag{4.8}$$

[2] In MD, if the center-of-mass frame is chosen, the position distribution is not uniform.

where n_{sample} denotes the total number of sampled configurations, and n_{ij} denotes the total number of disks among all the sampled configurations that appear in the bin (i, j). Clearly, $\hat{p}_{ij}$ is an unbiased estimator of Prob_{ij} [119]. The n_{sample} is conventionally a finite number. However, we will show that n_{sample} can be as much as T, the total number of transitions in the Markov chain that diverges in a continuous-time sampling algorithm.

The marginal probability density function is obtained by integrating $\pi^{(1)}(\mathbf{x}_i)$ over one direction

$$\pi^{(1)}{}_x(x) = \int_0^{L_y} \pi^{(1)}(x, y)dy. \tag{4.9}$$

Similarly, it is discretized by dividing the interval $[0, L_x]$ into n_b bins, each of size $\Delta x = L_x/n_b$. The probability of having a disk in the bin i, namely $[(i-1)\Delta x, i\Delta x]$, is

$$\text{Prob}_{x,i} = \int_{(i-1)\Delta x}^{i\Delta x} \pi^{(1)}{}_x(x)dx. \tag{4.10}$$

The associated observable is

$$p_{x,i}(\mathbf{x}) = \sum_{k=1}^{N} \Theta_{x,i}(x_k)/N, \tag{4.11}$$

where

$$\Theta_{x,i}(x_k) = \begin{cases} 1, & (i-1)\Delta x < x_k < i\Delta x; \\ 0, & \text{otherwise.} \end{cases} \tag{4.12}$$

The corresponding estimator is

$$\hat{p}_{x,i} = \frac{n_i}{N n_{\text{sample}}}, \tag{4.13}$$

where n_i denotes the total number of disks that appears in the bin i in all sampled configurations. The value of $\pi^{(1)}{}_x(x)$ in the bin i is approximated by $\hat{p}_{x,i}/\Delta x$. To obtain a better approximation, one can approximate the value of $\pi^{(1)}{}_x((i-1/2)\Delta x)$ by $\hat{p}_{x,i}/\Delta x$, and obtain the value of $\pi^{(1)}{}_x(x)$ at other points by fitting using the values at $(i-1/2)\Delta x$.

The probability density function scales implicitly as $O(1/V)$, as integrating this function over the whole box yields one. Consequently, the marginal probability density function scales implicitly as $O(1/L_x)$. It is convenient to define the rescaled probability density function that removes the scaling. Let

$$\rho(\mathbf{x}_i) = V\pi^{(1)}(\mathbf{x}_i), \tag{4.14}$$

and

$$\rho_x(x) = L_x\pi^{(1)}{}_x(x). \tag{4.15}$$

The rescaled probability density function $\rho(\mathbf{x})$ normalizes to V, and the rescaled marginal probability density function normalizes to L_x. The rescaled probability density functions appear in the pressure computation discussed in section 4.3.

The marginal position distribution can also be described by the cumulative distribution function, defined as

$$\Pi^{(1)}(x) = \int_{-\infty}^{x} \pi^{(1)}{}_x(x')dx'. \tag{4.16}$$

The cumulative distribution function needs no discretization. Define an observable

$$\mathit{\Pi}(\mathbf{x}; x) = \frac{\sum_{i=1}^{N} I_{[0,x]}(x_i)}{N}, \tag{4.17}$$

where I is the indicator function [119]. The expectation of this observable is $\mathit{\Pi}^{(1)}(x)$. The corresponding estimator is the empirical distribution function, defined as

$$\hat{\mathit{\Pi}}(x) = \frac{\sum_{k=0}^{n_{\text{sample}}-1} \sum_{i=1}^{N} I_{[0,x]}(x_i(t_k))}{n_{\text{sample}} N}, \tag{4.18}$$

The asymptotic behavior of the empirical distribution function is constrained by the DKW inequality [27, 79]. The empirical distribution always interpolates between 0 and 1, thus free of scaling with respect to V (L_x). The empirical distribution is used when comparing two sampling algorithms (discussed in detail in chapter 6).

4.2.1.2 Extrapolation of hard-disk position distribution

Here, we discuss how the precise estimation of the rescaled probability density function is performed at $x = \sigma$ and $x = L_x - \sigma$, where the probability density function is no longer continuous and reaches zero for a non-periodic box. These estimations are obtained by extrapolation, conventionally using a fourth-order polynomial [29]. In practice, the extrapolation is only performed in the neighborhood of $x \gtrapprox \sigma$ and $x \lessapprox L_x - \sigma$. The histogram is built for disks whose distances to the wall are less than 0.1σ. The interval of 0.1σ is further divided into 100 bins to build the histogram. In order to use the sampled configurations efficiently, disks near both walls at $x = 0$ and $x = L_x$ are taken into account. The number of disks in the bin $(0.1\sigma, 0.101\sigma)$ is the number of disks whose distance between either of the walls falls in the interval of $(0.1\sigma, 0.101\sigma)$. Although the histogram is built for only a small fraction of all sampled distances, the probability of having a disk in each of the bins is obtained by normalizing the histogram by dividing a factor of $2Nn_{\text{sample}}$. The probability density function is obtained by further dividing the normalized histogram by the bin size, and the 100 values obtained from the 100 bins, each associated to each center of the bins, are fitted by a fourth-order polynomial. In the attached publication 3, we use an estimator of the pressure which relies on the density at the wall, and that we develop an estimator that requires absolutely no binning. This method is essentially having a bin of vanishing size at $x = \sigma$, and the discussion of normalization in this section inspires the derivation of the estimator.

4.2.2 Pair-correlation functions

The rescaled probability density function of the relative position between two arbitrary disks is referred to as the pair-correlation function. Similar to the position distribution, it also appears in pressure computations.

Define the 2D pair-correlation function as $g(\mathbf{r}) = V\hat{g}(\mathbf{r})$ so that $\int g(x, y)dxdy = V$, where $\hat{g}(\mathbf{r}) = \hat{g}(x, y)$ is the probability density function such that, for a disk located at (x_0, y_0), the probability of finding a disk at $[x_0 + x, x_0 + x + dx] \times [y_0 + y, y_0 + y + dy]$ is $\hat{g}(x, y)dxdy$. Similar to the position distribution, $g(\mathbf{r})$ can be approximated by a histogram, which we do not elaborate on.

The probability of finding a pair of disks whose distance is smaller than r is

$$G(r) = \frac{1}{V}\int_0^r \int_0^{2\pi} g(\mathbf{r}')r'd\theta dr'. \tag{4.19}$$

The pair-correlation function $g(r)$ satisfies

$$G(r) = \frac{1}{V}\int_0^r \int_0^{2\pi} g(r')r'd\theta dr' = \frac{2\pi}{V}\int_0^r g(r')r'dr', \tag{4.20}$$

leading to the definition of the radial pair-correlation function

$$g(r) = \frac{V}{2\pi r}\frac{dG(r)}{dr}. \tag{4.21}$$

The relation (4.21) is how practically $g(r)$ is obtained. Let the bin size be Δr. The probability of finding a pair of disks having a distance falling in the bin i, namely $[(i-1)\Delta r, i\Delta r]$, is $\text{Prob}_{r,i} = G(i\Delta r) - G\left((i-1)\Delta r\right)$. Define an observable

$$p_{r,i}(\mathbf{x}) = \frac{2\sum_{i=1}^{N}\sum_{j=i+1}^{N} I_{[(i-1)\Delta r, i\Delta r]}(|\mathbf{x}_i - \mathbf{x}_j|)}{N(N-1)}. \tag{4.22}$$

The expectation of this observable is $\text{Prob}_{r,i}$, and the corresponding estimator is

$$\hat{p}_{r,i} = \frac{2\sum_{k=0}^{n_{\text{sample}}-1}\sum_{i=1}^{N}\sum_{j=i+1}^{N} I_{[(i-1)\Delta r, i\Delta r]}(|\mathbf{x}_i(t_k) - \mathbf{x}_j(t_k)|)}{n_{\text{sample}}N(N-1)}. \tag{4.23}$$

The value of $g(r)$ at $r = (i-1/2)\Delta r$ is approximated by $\frac{V\hat{p}_{r,i}}{2\pi(i-1/2)\Delta r}$, and the values at other points are approximated by a polynomial fit.

The value of $g(r)$ when two disks are at contact, namely $g(2\sigma)$, appears in pressure computation. It is approximated by extrapolating $g(r)$ computed from the histogram of distances in the neighborhood of $r \gtrapprox 2\sigma$. Conventionally, the histogram is built only using disks with distances less than 2.1σ and with bin size 0.001σ. The number of disks in each bin is counted, then divided by $n_{\text{sample}}N(N-1)\pi r_i/V$, where r_i is the distance at the center of each bin. Then, the value of $g(2\sigma)$ is obtained by extrapolating the histogram using a fourth-order polynomial. The pressure estimator depending on $g(2\sigma)$ has been first used by Metropolis et al. and remains as a primary method of obtaining pressure in hard-disk Markov-chain Monte Carlo computation for a long time. In the attached publication 3, we have developed an estimator of the pressure which relies on the pair-correlation function. Combined with the continuous sampling methods, the extrapolation and binning can be avoided, thanks to the infinite number of sampled configurations.

4.3 Pressure

In this section, we present the definitions of pressure for hard disks in a box, which can be periodic or have hard walls. Also, we derive the estimators for pressure by removing a piece of the box in various ways. Depending on the sampling algorithm, some of the estimators reduce into simple formulas. Although pressure is estimated for decades, we still manage to find new formulas that rely on the continuous-time nature of ECMC and molecular dynamics.

4.3.1 Definitions of pressure

Pressure can be either a quantity in mechanics or statistical physics. Depending on the context, pressure has two different definitions. However, in a non-periodic box, the different definitions of pressure lead to the same estimator of pressure and are equivalent. For a box of general aspect ratio, the pressures in the x direction and the y direction are different. The direction-free pressure is defined as the average of the two.

4.3.1.1 Kinematic definition

For the hard-disk model, the pressure is the force exerted per unit length by the hard disks on the wall. Since the collision is elastic and the disks are rigid, the force during the collision diverges. However, we will show that the time average of the force is finite. For a hard disk i traveling toward a wall located at L_x with velocity $v_{i,x}$, according to Newton's second law, the force, expressed as a function of time, is

$$F(t) = -\frac{2mv_{i,x}(t)}{dt} = -2mv_{i,x}\delta(t - t_c), \tag{4.24}$$

where t_c denotes the time of the collision. Instead of writing force as a delta function, it is possible to express (4.24) by the integral form of Newton's second law, namely

$$-2mv_{i,x} = \int_{t_c-\epsilon}^{t_c+\epsilon} F(t)dt, \tag{4.25}$$

where ϵ is a small time interval guarantees that there is only one collision between $[t_c-\epsilon, t_c+\epsilon]$. Summing over all of the collisions on the wall, one obtains

$$-2m \sum_{w:(i,+\hat{\mathbf{e}}_x)} v_{i,x} = \int_0^{t'} F(t)dt. \tag{4.26}$$

where $w : (i,+\hat{\mathbf{e}}_x)$ denote the collision of disk i with the wall in the x direction when traveling towards the wall at L_x, and t' is the total time of counting collisions. We are interested in the pressure at equilibrium, which remains as a constant over time and can be measured in experiments. The pressure in the x direction $P_x = -FL_y$ is thus treated as a constant.[3] Inserting (4.26) and expressing the force by its time average, one obtains an expression of the pressure P_x:

$$P_x = \frac{2m}{t'L_y} \sum_{w:(i,+\hat{\mathbf{e}}_x)} v_{i,x}. \tag{4.27}$$

For a square system, due to the symmetry of flipping the system with respect to the line $y = x$, one has $P_x = P_y$. In general, $P_x \neq P_y$. Also, this definition of pressure requires that there are walls in the system. Thus, it is not applicable to the periodic systems.

[3]The force exerted by the disk on the wall has a different sign than the force exerted by the wall on the disks.

4.3.1.2 Statistical-mechanics definition

In statistical mechanics, the pressure is defined as the response of Helmholtz free energy to the change of the volume, namely

$$\beta P = \frac{\partial \ln Z}{\partial V}, \tag{4.28}$$

also written as

$$\beta P = \lim_{\Delta V \to 0} \frac{1 - \frac{Z(V-\Delta V)}{Z(V)}}{\Delta V}. \tag{4.29}$$

This expression has the following interpretation: pressure is the fractional change of the partition function when removing a small piece of the system divided by the size of the small piece. Removing a small piece of the system naturally introduces a transform of the configuration. As the volume of the system change, some configurations are eliminated due to either overlap or having disks in the removed piece, leading to changes in free energy. However, there are various ways of removing the small piece, and we will demonstrate that how to reduce the volume is relevant for the value of the pressure. The detail of changing volume will be discussed in detail in section 4.3.2. Some possible ways of changing the volume is sketched in Fig. 4.1.

For hard disks, the temperature is not relevant and the energy of all allowed configurations are the same. So the partition function simply measures the "number" of configurations. And $1 - \frac{Z(V-\Delta V)}{Z(V)}$ has a straightforward interpretation, that is, the fraction of configurations that become invalid after removing a small piece of the system. Define an observable

$$P^*_{\Delta V}(\mathbf{x}) = \begin{cases} \frac{1}{\beta \Delta V}, & \text{configuration } \mathbf{x} \text{ is eliminated after the volume reduction;} \\ 0, & \text{configuration } \mathbf{x} \text{ remains valid after the volume reduction.} \end{cases} \tag{4.30}$$

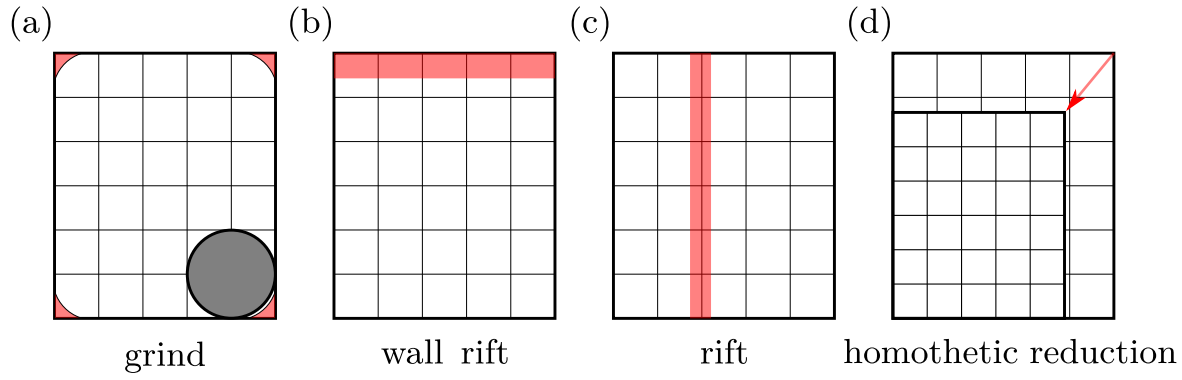

Fig. 4.1 *Volume reductions for a non-periodic $L_x \times L_y$ box. (a) Reducing V from the corners eliminates no samples, for sufficiently large σ. (b) Taking out a horizontal wall rift to estimate the wall density $\eta_y(L_y - \sigma)$ and the pressure P_y. (c) Taking out a vertical rift to estimate P_x. (d) Homothetic reduction of the box. Clearly, the pressure obtained as demonstrated in (a) is zero. (Figure from [71].)*

It is readily verified that the expectation of $P^*_{\Delta V}$ is P after taking the limit of $\Delta V \to 0$. The pressure estimator is thus

$$\beta\hat{P} = \lim_{\Delta V \to 0} \frac{\sum_{t=0}^{T-1} \beta P^*_{\Delta V}(\mathbf{x}(t))}{T}. \tag{4.31}$$

The derivation of the pressure estimator relies on choosing a volume-reduction procedure and identifying configurations to be eliminated. The trajectory $\mathbf{x}(t)$ of the sampling algorithm also plays a role in the derivation.

We have shown that, for a non-periodic box, certain ways of changing the volume lead to the equivalence between the statistical-mechanics pressure and the kinematic pressure.

4.3.2 Pressure estimators

In the present subsection, we reduce the volume by removing rifts, averaging over rifts, and uniformly shrinking the box and derive corresponding pressure estimators. We also derive pressure estimators by considering the momentum exchange with the wall and through the virial formula. Both derivations yield the same pressure estimator, indicating that the pressure is well-defined, and the discrepancies in the historic pressure computation are unrelated to the difference in the pressure definitions. The pressure estimators derived in this section are then used for high-precision pressure computations.

4.3.2.1 Wall rifts

Reducing the volume by rifts is first discussed in [88]. In this section, we discuss the straightforward alternative derivation we provide in the attached publication 3. Removing wall rifts is not the most efficient way of reducing the volume, but the equivalence of two definitions of pressure is shown via the wall-rifts pressure estimator.

Wall rifts refer to removing one slice at the boundary of a non-periodic box, as shown in Fig. 4.1(b). In an $L_x \times L_y$ box, the volume may be reduced through a vertical "rift" $[0, \epsilon] \times [0, L_y]$ with disk positions transforming as:

$$\{x, y\} \to \begin{cases} \emptyset & \text{if } x < \sigma + \epsilon \\ \{x - \epsilon, y\} & \text{if } x \geq \sigma + \epsilon, \end{cases} \tag{4.32}$$

where "$\emptyset$" means that the disk is eliminated. To motivate the much more evolved estimators that we discussed in detail in attached publication 3, we start with the naive "binning" approach discussed in section 4.2.1. The pressure can be estimated for finite ϵ from a finite number of samples, but then requires an extrapolation towards $\epsilon \to 0$. According to section 4.3.1.2, one can calculate pressure by finding the probability of being eliminated. This probability is approximated by the histogram, and the probability of eliminating a configuration by removing a rift is obtained by building a histogram of bin size $\epsilon \to 0$ and finding the probability of being in the bin located at $x = \sigma$ or $x = L_x - \sigma$. The result, as a function of ϵ, is summarized in Table 4.1. As $\epsilon \to 0$, the pressure approaches the reference value 10.7963 shown in Table I in the attached publication 3 [70]. The discrepancy is due to having multiple disks in the wall rift.

ϵ	βP
0.1	6.968543(3)
0.01	10.3242(2)
0.001	10.7676(6)

Table 4.1 – Thermodynamic pressure estimations for four disks of radius $\sigma = 0.15$ in a non-periodic square box of sides 1 using wall rift at $L_x - \epsilon$ of width ϵ. The error bar is obtained from 10 independent runs starting from the same initial configuration. As $\epsilon \to 0$, the pressure approaches the reference value shown in Table I in the attached publication 3 [70], which is obtained with the same settings and which is obtained without binning.

Following the derivation in the limit of $\epsilon \to 0$ in section III.C.1 in the attached publication 3 [70], one obtains the wall-rift pressure estimator, expressed in the rescaled probability density function:

$$\beta \begin{bmatrix} \hat{P}_x \\ \hat{P}_y \end{bmatrix} = \frac{N}{V} \begin{bmatrix} \rho_x(L_x - \sigma) \\ \rho_y(L_y - \sigma) \end{bmatrix}. \tag{4.33}$$

The boundary value of the rescaled line densities is obtained following the procedure discussed in section 4.2.1.

In EDMD and ECMC, the extrapolation can be avoided because of the infinite number of samples produced in a given run-time interval τ_{sim}. The sampling algorithms are described by Markov chains in the limit that the displacement of the active disk (or all disks in molecular dynamics) at each step $|d\mathbf{x}_i| = v_i dt$ is infinitesimal. For a run-time interval τ_{sim}, the number of steps, also the number of configurations that can be sampled, is

$$n_{\tau_{\text{sim}}} = \lim_{dt \to 0} \frac{\tau_{\text{sim}}}{dt} \propto \tau_{\text{sim}}. \tag{4.34}$$

In a continuous sampling algorithm, the time plays the role of the number of sampled configurations. For a distance interval Δx, and for one of the moving disks at constant velocity v, the number of sampled configurations is

$$n_{\Delta x} = \lim_{dt \to 0} \frac{\Delta x}{v dt} \propto \frac{\Delta x}{v}. \tag{4.35}$$

For convenience, the unit of configuration count is set to $1/dt$, and the number of sampled configurations is τ_{sim} during τ_{sim}, and $\Delta x/v$ for displacement Δx.

In EDMD, the rescaled line densities of eq. (4.33) can be computed, without extrapolation, by calculating the time spent by a disk when it is in the rift (the detailed discussion about the rift and wall collision in molecular dynamics can be found in section III.C.1 in the

attached publication 3 [70]). The resulting EDMD wall-rift estimator takes the form of:

$$\beta\hat{P}_x = \frac{1}{2L_y\tau_{\text{sim}}} \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} \frac{2}{|v^{\perp}_{(i)}|} \tag{4.36a}$$

$$= \left\langle \frac{2}{|v^{\perp}_{\text{wall}}|} \right\rangle \overbrace{\frac{1}{2L_y\tau_{\text{sim}}} \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} 1}^{\hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}}} \tag{4.36b}$$

$$= \frac{2\sqrt{\pi}}{\sqrt{\sum v_i^2}} \frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}} \tag{4.36c}$$

$$\overset{N\to\infty}{\longrightarrow} \sqrt{2\pi\beta m}\hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}}. \tag{4.36d}$$

The sum in eq. (4.36a) goes over the wall collisions w of all disks i in $\pm\hat{\mathbf{e}}_x$ direction, and $\hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}}$ is the wall-collision rate per vertical unit line element. Each wall collision is an indication of sampling configurations that have at least a disk close to the wall. The number of configurations having a disk in a fictive bin located at $x = \sigma$ is proportional to the inverse of its velocity in the x direction. In eq. (4.36a), $2/|v^{\perp}_{\text{wall}}|$ has an infinite variance. According to the velocity distribution (2.20), $2/|v^{\perp}_{\text{wall}}|$ has a power-law tail of exponent -3. However, with the analytical expression of the velocity distribution in section 2.1.2.1, this divergence is avoided by finding the expectation of $2/|v^{\perp}_{\text{wall}}|$ and turning the sum into collision counting. The pressure estimator of eq. (4.36c) can also be derived as a kinematic pressure estimator through the momentum transfer with the walls (see section 4.3.2.4). Thermodynamic and kinematic pressures thus agree already at finite N.

The same argument applies to straight ECMC (see section III.C.1 in the attached publication 3 [70] for a detailed derivation). ECMC only detects the presence of the active disk in the wall rifts, and a configuration to be eliminated is detected with a probability biased by a factor of $1/N$. This bias is corrected by multiplying the right-hand side of eq. (4.36a) by N, resulting in the ECMC wall-rift estimator:

$$\beta\hat{P}_x = \frac{N}{2L_y\tau_{\text{sim}}} \sum_{w:(a,\pm\hat{\mathbf{e}}_x)} \frac{2}{|v^{\perp}_{\text{wall}}|} = 2N\hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}} \tag{4.37}$$

The same bias-correcting factor also applies to the reflective, forward, Newtonian variant of ECMC. The derivation of the contribution at each wall collision in these variants is identical to the derivation of EDMD, and the bias-correction factor N is still needed. The pressure estimator for the reflective, forward, and Newtonian ECMC is

$$\beta\hat{P}_x = \frac{N}{2L_y\tau_{\text{sim}}} \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} \frac{2}{|v^{\perp}_{\text{wall}}|}. \tag{4.38}$$

The derivation of (4.38) uses exactly the same idea as the derivation of (4.36a). The number of the eliminated configurations after the transform is proportional to the inverse of the active disk's velocity in the x direction. The distribution of velocity for the three variants at the wall is not known, and there is no further derivation rooted from eq. (4.38). Numerical experiment shows that the distribution of $2/|v^{\perp}_{\text{wall}}|$ has a power-law tail of exponent -3,

thus a diverging variance. Although a diverging variance may result in persisting bias in stochastic computation [62], we have not observed such bias.

4.3.2.2 Rift averages

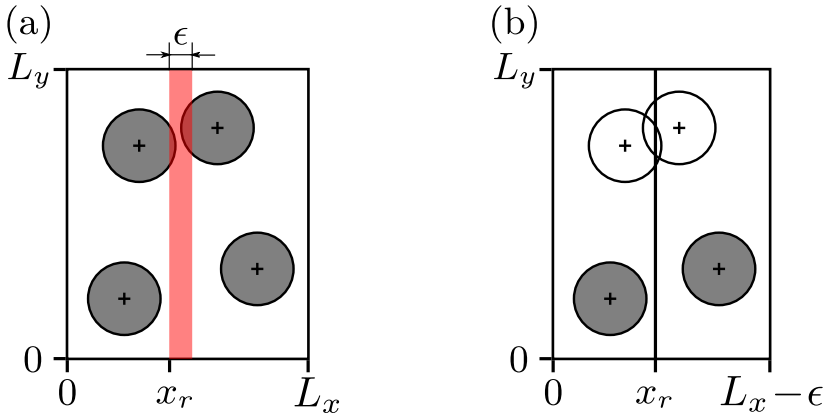

Fig. 4.2 *Vertical rift* $[x_r, x_r+\epsilon]\times[0,L_y]$. *(a): $L_x\times L_y$ box with vertical rift of width ϵ at position x_r. (b): Transformed sample, which is eliminated because of a pair overlap. (Figure from* [70].)

In an $L_x\times L_y$ box, the volume may be reduced through a vertical rift inside the box $[x_r, x_r+\epsilon]\times[0,L_y]$ with disk positions transforming as:

$$\{x,y\}\to\begin{cases}\{x,y\} & \text{if } x<x_r\\ \emptyset & \text{if } x_r\le x<x_r+\epsilon\\ \{x-\epsilon,y\} & \text{if } x\ge x_r+\epsilon.\end{cases} \tag{4.39}$$

The pressure should not depend on where the rift is removed. Thus, the pressures obtained by removing a rift at any possible $x_r\in[0,L_x]$ are the same. Averaging these pressures yields the same pressure as each of them, and, the average uses sampled configurations more efficiently. The wall rift can eliminate a configuration only if there is a disk close to the wall (for finite ϵ) or a wall collision (for $\epsilon\to 0$). Similarly, the pressure estimator obtained by removing a random rift considers configurations possessing two disks close enough to each other at the rift or has a disk in the rift (for finite ϵ). To use the sampled configurations more efficiently, one can remove the rift at every possible place and average the corresponding pressure estimators. In this way, as long as a configuration have two disks close to each other, it contributes to the pressure estimator. The resulting estimator is the rift-average pressure estimator.

The eliminated configurations having disks in the rift (demonstrated in Fig. 4.3) lead to the ideal-gas contribution to the pressure estimator:

$$\beta\hat{P}_x^{\text{ideal gas}}=\frac{\epsilon N}{L_x}\frac{1}{\epsilon L_y}=\frac{N}{V}. \tag{4.40}$$

This term is always in the rift-average pressure estimator, even if the radius of the disk is zero.

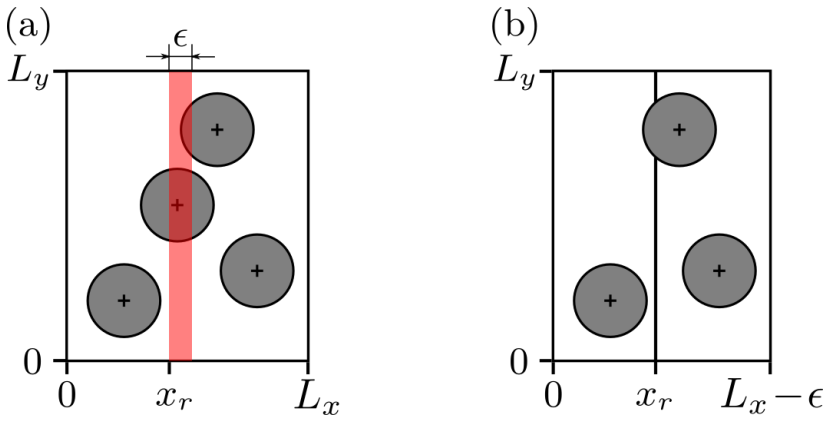

Fig. 4.3 *Having a disk in the rift* $[x_r, x_r+\epsilon]\times[0, L_y]$. *In the limit of* $\sigma \to 0$, *configurations can only be eliminated in this way. The contribution of these configurations to pressure is thus identical to the ideal gas pressure.* (*a*): $L_x \times L_y$ *box with vertical rift of width* ϵ *at position* x_r *and a disk whose center is in the rift.* (*b*): *Transformed sample, which is eliminated because of a missing disk.*

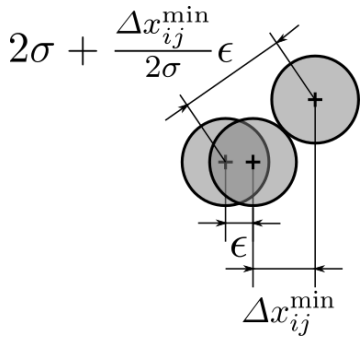

Fig. 4.4 *Two disks that are close to each other. When in contact, the distance between the disks in the* x *direction is* $\Delta x_{ij}^{\min}$. *Displacing one of the disks in the* x *direction such that removing a rift of width* ϵ *between the disks leads to contact, the distance between the centers of the disks is* $2\sigma + \frac{\Delta x_{ij}^{\min}}{2\sigma}\epsilon$ *at precision* $O(\epsilon)$.

Furthermore, the configuration is eliminated if the transformed configuration contains overlaps, indicated by disk collisions in the limit $\epsilon \to 0$. The overlap contribution of pressure is derived considering the sample shown in Fig. 4.2. When the distance in the x direction is $\Delta x_{ij}^{\min} + \epsilon$, where $\Delta x_{ij}^{\min}$ is the x-separation at contact, the distance between center of the disks is approximately $2\sigma + \frac{\Delta x_{ij}^{\min}}{2\sigma}\epsilon$. This is due to (demonstrated in Fig. 4.4)

$$(\Delta x_{ij}^{\min}+\epsilon)^2+4\sigma^2-(\Delta x_{ij}^{\min})^2 = 4\sigma^2+2\epsilon\Delta x_{ij}^{\min}+\epsilon^2 = \left(2\sigma + \frac{\Delta x_{ij}^{\min}}{2\sigma}\epsilon\right)^2 + \left(1 - \frac{(\Delta x_{ij}^{\min})^2}{4\sigma^2}\right)\epsilon^2, \tag{4.41}$$

indicating the distance to be $2\sigma + \frac{\Delta x_{ij}^{\min}}{2\sigma}\epsilon$ at precision $O(\epsilon)$. If there are disks i and j whose distance is less than $2\sigma + \epsilon\Delta x_{ij}^{\min}/(2\sigma)$, the configuration is eliminated. Let the relative velocity between two disks at collision be $\Delta \mathbf{v}_{\text{pair}}$, and its component on the line connecting

the disk centers at collision be $\Delta v^{\perp}_{\text{pair}}$. The time spent in to-be-eliminated configurations is $(2/|\Delta v^{\perp}_{\text{pair}}|)[\epsilon \Delta x^{\min}_{ij}/(2\sigma)]$. The factor 2 is due to that the two disks at collisions approach each other before the collision and leave each other after it. The configurations sampled during approaching contributes $(1/|\Delta v^{\perp}_{\text{pair}}|)[\epsilon \Delta x^{\min}_{ij}/(2\sigma)]$ to the pressure estimator, and so does configurations sampled during leaving in EDMD. The probability of having a rift between two disks is bounded by $\Delta x^{\min}_{ij}/L_x$ and $(\Delta x^{\min}_{ij} + \epsilon)/L_x$. Taking the limit $\epsilon \to 0$, only configurations having two disks colliding with each other contribute to the pressure estimator for an amount of

$$\frac{|\Delta x^{\min}_{ij}|^2}{2\sigma} \frac{2}{\Delta v^{\perp}_{\text{pair}}}. \tag{4.42}$$

Following the derivation in section III.C.1 in the attached publication 3 [70], one obtains the EDMD rift-average estimator for P_x:

$$\beta \hat{P}_x = \frac{N}{V} + \frac{1}{V\tau_{\text{sim}}} \left[\sum_{p:(ij)} \frac{|\Delta x^{\min}_{ij}|^2}{2\sigma} \left\langle \frac{2}{\Delta v^{\perp}_{\text{pair}}} \right\rangle + \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} \left\langle \frac{2\sigma}{|v^{\perp}_{\text{wall}}|} \right\rangle \right], \tag{4.43}$$

and for $P = (P_x + P_y)/2$:

$$\beta \hat{P} = \frac{N}{V} + \frac{\sigma}{V\tau_{\text{sim}}} \left[\sum_{p:(ij)} \left\langle \frac{2}{\Delta v^{\perp}_{\text{pair}}} \right\rangle + \sum_{w:(i,\pm\hat{\mathbf{e}}_x,\pm\hat{\mathbf{e}}_y)} \left\langle \frac{1}{|v^{\perp}_{\text{wall}}|} \right\rangle \right]. \tag{4.44}$$

In a non-periodic box, using the velocity distribution eqs (2.20) and (2.22), the EDMD rift-average estimator takes the form of

$$\beta \hat{P} = \frac{N}{V} + \frac{\sigma\sqrt{\pi}}{\sqrt{\sum v_i^2} V} \frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} (n_{\text{wall}}^{\pm\hat{\mathbf{e}}_x \pm\hat{\mathbf{e}}_y} + \sqrt{2} n_{\text{pair}}) \tag{4.45a}$$

$$\overset{N\to\infty}{\longrightarrow} \frac{N}{V} \left(1 + \frac{\sigma\sqrt{\pi m\beta}}{N} n_{\text{pair}} \right), \tag{4.45b}$$

where $n_{\text{wall}}^{\pm\hat{\mathbf{e}}_x \pm\hat{\mathbf{e}}_y}$ is the wall-collision rate, the number of all wall collisions per time interval, and similarly for the pair-collision rate n_{pair}. In the $N \to \infty$ limit of eq. (4.45b), wall collision play no role.

Rift-average pressure estimators for ECMC detect wall and pair collisions with biases (see eq. (4.37)), that must again be corrected, namely by a factor N for each wall event and by a factor $N/2$ for each pair event. The latter is because a lifted sample of N disks to be eliminated is detected only if either i or j are active (see Fig. 4.5c). The factor $N/2$ also has an alternative interpretation. There are $N(N-1)/2$ pairs of disks that can be close to each other, but only $1(N-1)$ can be detected as only a single active disk is moving. The factor is thus $N(N-1)/2$ divided by $1(N-1)$, yielding $N/2$. Taking into account the bias-correction factors, the straight-ECMC rift-average estimator is obtained:

$$\beta \hat{P}_x = \frac{N}{V} + \frac{N}{V\tau_{\text{sim}}} \left(\sum_{p:(ij)} \Delta x^{\min}_{ij} + \sum_{w:(i\pm\hat{\mathbf{e}}_x)} 2\sigma \right). \tag{4.46}$$

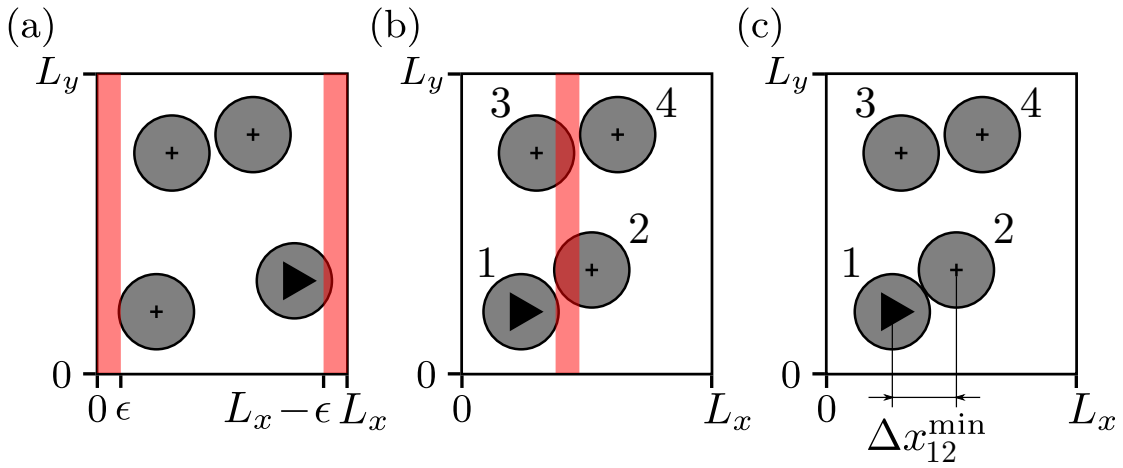

Fig. 4.5 *ECMC rift estimators. (a): The ECMC wall-rift estimator only detects rift overlaps of the active disk, explaining the factor N in eq. (4.37), that is absent in eq. (4.36a). (b): A pair of disks (i,j) leading to the elimination of the sample is detected only if either i or j are active, explaining a factor $N/2$ entering eq. (4.46). (c): Illustration of the x-separation at contact $\Delta x_{ij}^{\min}$ (also relevant for EDMD). (Figure from [71].)*

The details and alternative forms of this estimator can be found in section III.C.1 in the attached publication 3 [70]. In a periodic box, the wall collision terms in (4.46) is dropped, leading to

$$\beta\hat{P}_x = \frac{N}{V} + \frac{N}{V\tau_{\text{sim}}}\sum_{p:(ij)} \Delta x_{ij}^{\min}. \tag{4.47}$$

In practice, $\hat{P}_x$ is computed when the active disk moves in $\hat{\mathbf{e}}_x$, and $\hat{P}_y$ is computed when the active disk moves in $\hat{\mathbf{e}}_y$. The original rift formula proposed in [88] is the same as (4.47), and is has been much used. In the attached publication 3, we put its derivation onto a firm basis. As will be shown, there are also a number of generalizations that we are able to derive. In our practical calculations, the formula (4.47) can also be used even for simulation algorithms that produce discrete sets of samples. This is done by running tiny-chain-length straight ECMC in sampled configurations.

For multithreaded ECMC, as there are multiple active disks, the contact of $N_A(N-N_A)$ pairs of disks are detectable. The bias-correcting factor thus becomes $\frac{N(N-1)}{2N_A(N-N_A)}$, leading to the pressure estimator in multithreaded ECMC:

$$\beta\hat{P}_x = \frac{N}{V} + \frac{N(N-1)}{[N_A(N-N_A)]V\tau_{\text{sim}}}\sum_{p:(ij)} \Delta x_{ij}^{\min}. \tag{4.48}$$

We have derived the pressure formula of reflective, forward, and Newtonian ECMC, discussed in chapter 3, also by calculating the time spent in the rift. Each collision contributes

$$\frac{|\Delta x_{ij}^{\min}|}{2\sigma}\left(\frac{1}{\Delta v_{\text{in}}^{\perp}}+\frac{1}{\Delta v_{\text{out}}^{\perp}}\right) = \frac{|\Delta x_{ij}^{\min}|}{2\sigma}\left(\frac{1}{|v_{\text{in}}^{\perp}|}+\frac{1}{|v_{\text{out}}^{\perp}|}\right) \tag{4.49}$$

to the pressure estimator, where $v_{\text{in}}^{\perp}$ and $v_{\text{out}}^{\perp}$ denote the velocity of the active disk when the active disk approaches its collision partner and the new active disk leaves the stopped active

disk, respectively. Here, the relative velocity is substituted by the absolute value of velocity, since, in ECMC, one of the disks in collision is always at rest. The configurations sampled when the old active disk approaches the target disk contribute the term proportional to $1/|v_{\text{in}}^{\perp}|$, and the ones sampled when the new active disks leaving the stopped active disk contribute the term proportional to $1/|v_{\text{out}}^{\perp}|$. As in (4.46), there is a factor $N/2$ to correct the bias for disk collision. The contribution at the wall is identical to EDMD when there is a collision at the wall. Summing over all the contributions, one finds

$$\beta \hat{P}_x = \frac{N}{V} + \frac{N}{V\tau_{\text{sim}}} \left[\sum_{p:(ij)} \frac{|\Delta x_{ij}^{\min}|^2}{4\sigma} \left\langle \frac{1}{|v_{\text{in}}^{\perp}|} + \frac{1}{|v_{\text{out}}^{\perp}|} \right\rangle + \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} \left\langle \frac{2\sigma}{|v_{\text{wall}}^{\perp}|} \right\rangle \right]. \tag{4.50}$$

The distribution of velocity at collision is unknown, so this formula cannot be further reduced. Computation shows that the distribution of $1/|v_{\text{in}}^{\perp}|$ and $1/|v_{\text{out}}^{\perp}|$ has a power-law tail of scaling -3, indicating a diverging variance.

4.3.2.3 Homothetic volume reductions

Besides by rifts, the volume V of an $L_x \times L_y$ box can be reduced by a homothetic transformation, where the box and all positions $\mathbf{x}_i$ are homogeneously scaled by a factor $1-\epsilon_\alpha < 1$, while the disk radii σ remain unchanged. The transformation of the box corresponds to simultaneous removing horizontal and vertical rifts of equal rift volume, but the disk positions then transform inhomogeneously, as in eq. (4.39).

A homothetic volume reduction yields the pressure $\beta P = \beta(P_x + P_y)/2$, rather than P_x or P_y. It may be performed in two steps. In a first step (from (σ, V) to (σ', V), see Fig. 4.6), the box and the $\mathbf{x}_i$ are unchanged, but the disks are swollen by a factor $1/(1-\epsilon_\alpha)$, possibly eliminating samples. In a second step, all lengths are rescaled by $1-\epsilon_\alpha$, so that the radii return to σ. This second step (from (σ', V) to (σ, V')) is rejection-free, and its reduction of sample-space volume, with $Z(\sigma, V') = (V'/V)^N Z(\sigma', V)$, constitutes the ideal-gas term of the pressure.

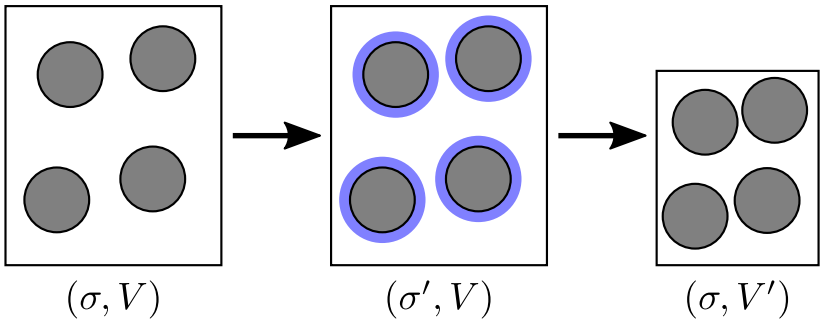

Fig. 4.6 *A homothetic volume reduction performed through a swelling of disks followed by a uniform reduction of all lengths* (*box, positions, radii*). (*Figure from* [70].)

In the two-stage transition $Z(\sigma, V) \to Z(\sigma', V) \to Z(\sigma, V')$, the two-step procedure turns

eq. (4.29) into

$$\begin{aligned}\beta P \overset{V'\to V}{=} & \frac{\log\left[Z(\sigma,V)\right]-\log\left[Z(\sigma,V')\right]}{V-V'}\\ =& \frac{N}{V}+\frac{1}{V-V'}\frac{Z(\sigma,V)-Z(\sigma',V)}{Z(\sigma,V)}.\end{aligned} \tag{4.51}$$

The final term again divides the elimination probability of a sample by the change of volume (see also [3, 20, 24, 30]).

The pair-elimination probability is related to the pair-correlation function, and the wall-elimination probability is related to the rescaled line density. Following the derivation in section III.C.2 in the attached publication 3 [70], the pressure estimator is obtained:

$$\beta\hat{P} = \frac{N}{V}+\frac{N}{V}\left[2\pi\frac{(N-1)\sigma^2}{V}g(2\sigma)+\sigma\frac{\rho_x(\sigma)}{L_x}+\sigma\frac{\rho_y(\sigma)}{L_y}\right] \tag{4.52a}$$

$$\overset{N\to\infty}{\longrightarrow} \frac{N}{V}\left[1+2\eta g(2\sigma)\right]. \tag{4.52b}$$

Eq. (4.52b) has long been used for estimating pressures in MCMC [84].

EDMD and ECMC can estimate the pressure without extrapolations of the pair correlation functions and the wall densities by tracking the time during which close pairs exist, or a disk is close to the wall. This simply reproduces eqs (4.44) and (4.45) for EDMD. The corresponding homothetic pressure estimators for all variants of ECMC are readily derived, but they have diverging variances. Besides, as the active disk cannot detect the wall in the y direction while moving in the x direction in straight ECMC, the homothetic pressure estimator in straight ECMC can only be applied to a periodic box.

4.3.2.4 Kinematic pressure estimators

The kinematic pressure estimators, implemented in EDMD, is derived based on the realistic dynamics of hard disks. A wall estimator is derived following the kinematic definition of pressure:

$$\hat{P}_x = \frac{1}{2L_y\tau_{\text{sim}}}\sum_{w:(i,\pm\hat{\mathbf{e}}_x)} 2m|v^{\perp}_{\text{wall}}| \tag{4.53a}$$

$$= 2m\left\langle|v^{\perp}_{\text{wall}}|\right\rangle \hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}} \tag{4.53b}$$

$$= \frac{mR\sqrt{\pi}}{N}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)}\hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}}, \tag{4.53c}$$

$$\overset{N\to\infty}{\longrightarrow} \sqrt{2\pi\beta m}\hat{n}^{\pm\hat{\mathbf{e}}_x}_{\text{wall}}, \tag{4.53d}$$

where in eq. (4.53c), we used the analytical expectation of velocity eq. (2.20b). At finite N, the estimator eq. (4.53c) is identical to the wall-rift pressure estimator of eq. (4.36c) derived from the thermodynamic definition of pressure. Thus, different definitions of pressure lead to the same estimator and the same numerical result.

An alternative estimator can be derived from the virial function

$$G_x = m\sum_{i=1}^{N} x_i v_{i,x}, \tag{4.54}$$

which is strictly bounded during molecular dynamics. The time average of its time derivative is thus bounded by $|G_x|/t'$, which is zero in the limit that the run time $t' \to \infty$. Thus, we have

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}t} G_x \right\rangle = m \left\langle \sum_{i=1}^{N} (x_i \dot{v}_{i,x} + v_{i,x}^2) \right\rangle = 0. \tag{4.55}$$

The derivation of velocity can be expressed in terms of the momentum exchange, thus associated to collisions. The exact expression of each terms in (4.55) is obtained by tracking the momentum exchange at collisions. Following the detailed derivation discussed in section III.C.3 in the attached publication 3 [70], a kinematic EDMD pressure estimator is derived from (4.55)

$$\beta \hat{P}_x = \frac{N}{V} + \frac{\beta}{V \tau_{\text{sim}}} \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} \left\langle 2\sigma |v_{\text{wall}}^{\perp}| \right\rangle + \sum_{p:(ij)} \frac{(\Delta x_{ij}^{\min})^2}{4\sigma^2} \left\langle 2\sigma \Delta v_{\text{pair}}^{\perp} \right\rangle. \tag{4.56}$$

The pressure is introduced into (4.55) by relating the pressure to wall collision using (4.53). Inserting the velocity distribution eqs (2.20) and (2.22), this kinematic estimator becomes equivalent to the thermodynamic rift-average estimator of eq. (4.43).

The distribution of velocity plays a crucial role in the molecular dynamics, in particular, in the pressure calculation using kinematic pressure estimators. As discussed in section 3.2.4, the molecular dynamics samples a NVE**MR** ensemble for a system implementing a periodic box. In this case, an extra factor of $\frac{N-1}{N}$ appears in the pressure estimator, leading to an expression of

$$\beta \hat{P}_x = \frac{N}{V} + \frac{N-1}{N} \sum_{p:(ij)} \frac{(\Delta x_{ij}^{\min})^2}{4\sigma^2} \left\langle 2\sigma \Delta v_{\text{pair}}^{\perp} \right\rangle. \tag{4.57}$$

This formula is obtained in [31] and [130] using two different methods. The high precision computation in a four-disk system numerically confirms this formula. The factor $(N-1)/N$ vanishes in the thermodynamics limit, and the relative correction introduced by this factor is lower than 10^{-6} for $N = 1024^2$. So, for large systems, this factor can be ignored.

4.4 Orientational order of hard disks

The orientational order is recognized as an important aspect of physics in two-dimensional systems [82]. It reflects the position of a disk relative to its neighbors, namely the orientation of the line connecting a disk and its neighbor. The orientational order parameter was originally defined for triangular lattice in [40] as

$$\psi_6 = \mathrm{e}^{6i\phi}, \tag{4.58}$$

where ϕ is the angle of the line connecting an atom and its nearest neighbor relative to $\hat{\mathbf{e}}_x$. The factor 6 is due to the symmetry of rotating $\pi/3$ of the triangular lattice. The orientation between 0 and $\pi/3$ is mapped onto $[0, 2\pi]$ by the factor 6. As the fully packed configuration in the hard-disk model is arranged as a triangular lattice, the idea of orientational order is also applicable to the hard-disk model. The orientational order is a crucial observable for

hard-disk phase transition, as the hexatic phase is characterized by the algebraic decay of the orientational order.

For the hard disk model, the orientation order for each disk is defined as

$$\psi_6(l) = \frac{1}{\mathrm{nbr}(l)} \sum_{j=1}^{\mathrm{nbr}(l)} e^{6i\phi_{lj}}, \tag{4.59}$$

where nbr is the number of neighbors in the Voronoi diagram of disk l, and ϕ_{lj} is the angle of the line connecting disk l and its jth neighbor relative to $\hat{\mathbf{e}}_x$. This local orientational order can be used as a visual confirmation of the phase coexistence [10]. The global orientation is defined as the average of the local orientational order over all disks, namely

$$\Psi_6 = \frac{1}{N} \sum_l \psi_6(l). \tag{4.60}$$

Ψ_6 itself is defined in the form of an observable. Its estimator is the average of Ψ_6 over all sampled configurations. In a fluid, the system has no orientational order, and $\langle|\Psi_6|\rangle$ is close to 0. For a fully packed configuration possessing perfect orientational order, $\langle|\Psi_6|\rangle = 1$.

For a hard-disk ensemble, the probability density function of the configurations in the sample space can be translated to a probability density function of orientational order in the complex plane. The probability density possesses symmetries, inherited from the symmetries of the configurations. In a box with an arbitrary aspect ratio, flipping a configuration with respect to the x axis or y axis yields another configuration. As a consequence, any Ψ_6 and its complex conjugate have the same statistical weight, and the expectation of Ψ_6 must be on the real axis. In a square box, rotating a configuration by $\pi/2$ yields another configuration. The configuration containing hexagons pointing in $\hat{\mathbf{e}}_x$ is referred to as a "base" configuration, while the configuration containing hexagons pointing in $\hat{\mathbf{e}}_y$ is referred to as a tip configuration. A typical base configuration and a typical tip configuration are shown in Fig. 4.7. As a consequence, any Ψ_6 has the same statistical weight as $\Psi_6 \times \mathrm{e}^{i\pi}$. The complex plane is thus divided into four partitions, specified by $\frac{\pi}{2}i < \arg\Psi_6 < \frac{\pi}{2}(i+1), i = 0, 1, 2, 3$. The probability distribution of Ψ_6 in each of the regions can be obtained from the probability distribution in another region by rotation and reflection. Due to both symmetries, the expectation of global orientation is zero in a square box. Having a known expectation allows one to use the global orientation as a probe for the convergence of the Markov chain (see section 7.1.1).

The local orientational order is calculated following its definition by finding the angle of the line connecting a disk and its neighbors in the Voronoi diagram. In practice, the neighboring disks in the Voronoi diagram are found by Delaunay tessellation.

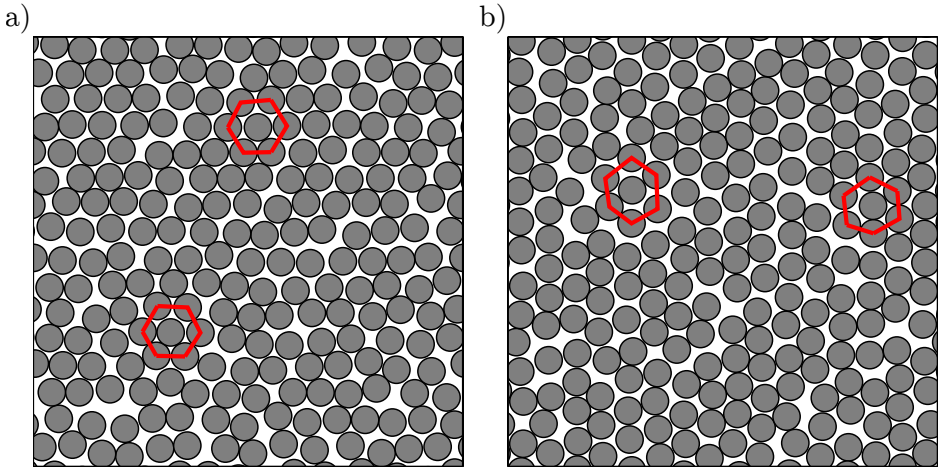

Fig. 4.7 *Two typical configurations for* $N = 224$, $\eta = 0.71$, *with red line highlighting the typical hexagons.* (a) *A typical base configuration whose* $\Psi_6 = 0.70 - 0.01i$. (b) *A typical tip configuration whose* $\Psi_6 = -0.73 + 0.01i$.

Chapter 5

Implementation of algorithms

Sampling algorithms and estimators (the first two steps in the stair-case structure of Fig. 1.1) are implemented in computer programs. The implementation process itself is complicated and merits separate attention. As an example, the velocity of a given disk after collisions in forward ECMC can be expressed as a single formula containing a random number. However, in the computer program, the random number is generated from a complicated procedure that features arithmetic and logical operations. The use of random numbers has become so common that one hardly realizes that it depends on a multitude of logicical operations. The gap between the abstract algorithm and its implementation is more apparent in our multithreaded ECMC introduced in chapter 3, in which we have implemented lock-free programming that may introduce random infrequent wrong results that are practically undetectable. Furthermore, The implementation problems posed by GPU programming stem from data parallelization, and are very different from the task parallelism in multithreaded ECMC. The problem regarding the multithreaded ECMC is solved by describing our implementation in the sequential consistency and mapping our implementation onto an absorbing Markov chain. So, in summary, an algorithm leads to an implementation, and again leads to an abstract algorithm, whose behavior can be proven.

In this chapter, we first present the formal verification of the computer programs, especially the sequential consistency we have used to verify our multithreaded ECMC program in C++. Our absorbing-Markov-chain analysis in the attached publication 1 is also discussed, serving as a case study of formal verification. We further discuss the performance of the implementation of sampling algorithms, especially the dependence of the performance of multithreaded ECMC on the hardware. The present chapter follows closely the section 2.3, 3.2, 3.3 in the attached publication 1 [71].

5.1 Formal verification

In multithreaded ECMC, both the position and the local time of an active disk are updated at collisions. However, today's computer architecture allows for only a single operation at a time on a single thread. Therefore, the specific order of updating the two attributes of the active disk has to be defined. Naturally a question arises: will any order of updates leads to the same output as specified by the sampling algorithm? This question is answered by the formal verification procedure, which refers to describing the implementation of the

computer program by mathematical models and proving rigorously the correctness of this formal implementation. In this section, we present our computer program for multithreaded ECMC in C++ in the framework of sequential consistency and discuss the design of the program and the language features we have implemented. Then we show that the order of updating position and local time can lead to differing outcomes, and that there is no deadlock in our program.

5.1.1 Sequential Consistency

Sequential consistency [65] is a tool for describing the causal relation between operations in multithreaded programs. It is a mathematical representation of a computer program. In the sequential consistency, the program is represented by a sequence of basic operations such as read, write, and arithmetic operations. The operations on each thread are executed by a given order. However, the order of operations on different threads is not specified, and there are numerous orders of executing operations on multiple threads (demonstrated in Fig. 5.1). An abstract operation in the sampling algorithm (for example, updating velocity) is usually realized through a collection of operations on a computer. The causal relation of two collections of operations falls in one of the two cases: "precedes" and "can affect". Let A and B be collections of operations. A precedes B means all operations in A precede all operations in B. In this case, B has no impact on A, as operations in A have already finished before executing the first operation in B. A can affect B means that some of the operations in A precede some of the operations in B, and it is hard to say whether A and B affect each other. The sequential consistency comes with six axioms that allow the derivation of the causal relation between any two collections of operations. For example, the mutual exclusion of a collection of operations C, i.e. only one of the threads is allowed to execute operations C at a time, can be proven via mathematical derivation showing that the operations C on different threads precede each other.[1] The sequential consistency takes into account that, when running multithreaded program on a real processor, the order of operations on different threads cannot be controlled, and that running a program repeatedly does not exhaust all possible orders. It solves the problem that sometimes an undesired behavior is too infrequent to be observed.

5.1.2 Example: Sequential consistency of multithreaded ECMC

The implementation of the multithreaded ECMC algorithm is abstracted as Algorithm 1. Operations 1ι to 20ι are executed repeatedly before the active disk associated to the thread covers a given distance. In each iteration, the thread first computes the time it takes for its active disk to collide with the three potential targets specified by the constraint graph $(2\iota,3\iota,4\iota)$. In the meantime, horizon violations, introduced in chapter 3, are checked for (5ι). A horizon violation terminates all threads. One of the three potential targets, which has the lowest time of flight, is identified as the target of collision $(6\iota, 7\iota, 8\iota, 9\iota)$. Then, the thread tries to acquire the target by a compare-and-swap (CAS) operation (10ι). Acquiring the target disk requires it being static. If it is successfully acquired, the thread checks whether the position of the target disk is updated since the time of flight was computed (13ι). If

[1] In multithreaded programs, the operations on each thread are usually implemented in loops. Thus, the collection of operations C can be executed repeatedly, both before and after specific operations on other threads.

it remains the same, the local time and the position of the active disk are increased by the time of flight $(15\iota, 16\iota)$, and the local time of the target disk becomes the same as the active disk (14ι). At last, the thread releases the active disk before the collision by setting its tag to static (17ι). If the target disk is active or its position has changed since the calculation of the time of flight, the thread directly enters the next iteration and starts from the time-of-flight computation.

Algorithm 1 (Multithreaded ECMC (abstraction of implementation)) *This algorithm is taken from [71]. The operations in this algorithm take place between two breakpoints. Before the run starts, all local times are set to 0, and all tags are put to static, except for the active spheres, whose tags correspond to their thread ι. The buffer is set to $\{1a, 1b\}$. A random switch selects one buffer element. The corresponding statement is executed on its thread, and the buffer is replenished. The following provides pseudo-code for the multithreading stage (i_ι is the active sphere, j_ι the target sphere, and $distance_\iota$ the difference between the chain length and the local time, all on thread ι):*

```
1ι   τ_ι ← distance_ι; j_ι ← i_ι; x_ι ← ∞
2ι   for j̃ in {1,2,...,n} \ i_ι :
3ι     x_j̃ ← j̃.x
4ι     τ_ij̃ ← x_j̃ − i_ι.x − b_ij̃
5ι     if i_ι.t+τ_ij̃<j̃.t : abort
6ι     if τ_ij̃ < τ_ι :
7ι       j_ι ← j̃
8ι       x_ι ← x_j̃
9ι       τ_ι ← τ_ij̃
10ι  j_ι.tag.CAS(static,ι)
11ι  if j_ι.tag = ι :
12ι    if τ_ι < distance_ι :
13ι      if x_ι = j_ι.x :
14ι        j_ι.t ← i_ι.t + τ_ι
15ι        i_ι.t ← i_ι.t + τ_ι
16ι        i_ι.x ← i_ι.x + τ_ι
17ι        i_ι.tag ← static
18ι        distance_ι ← distance_ι − τ_ι
19ι        i_ι ← j_ι
         else :
20ι        j_ι.tag ← static
           goto 1
       else :
21ι      i_ι.t ← i_ι.t + τ_ι
22ι      i_ι.x ← i_ι.x + τ_ι
23ι      distance_ι ← 0
24ι      i_ι.tag ← stalled
     else : goto 1
25ι  if distance_ι > 0 : goto 1
26ι  wait
```

When all k threads have reached their **wait** *statements, the algorithm terminates.*

The implementation is based on a data structure in which each disk is defined as an object, which contains the position, three arrows in the constraint graph, local time, and tag. The tag denotes whether the disk is active or static. The disk objects are stored together as an array. The multithreaded ECMC is implemented on a shared-memory multithreaded computer. The operations in Algorithm 1 are executed simultaneously on multiple threads (logical processors) which are labeled by their identity ι. The array storing the disk objects can be read from and written into by all threads, so each thread "sees" the work done by other threads in real-time.

A common issue with the shared-memory program is data racing, that is, multiple threads trying to write to the same location in the memory. Data racing leads to undefined behavior such that the involved memory location stores an unpredictable value. Data racing is avoided by forbidding two threads from moving a single disk at the same time. This is achieved by defining the tag as an "atomic" variable and using the atomic "compare-and-swap" operation to update its value. The tag of an active disk is the thread identity ι, and is stalled when the active disk reaches the next breakpoint. In other cases, the disk tag is static. At collisions, the thread (of identity ι) reads the tag of the target. If it is the identity of another thread (for example ι'), thread ι waits until thread ι' moves its active disk away. If it is static, the collision is allowed to take place, and thread ι writes its identity ι to the tag of the target. If the compare-and-swap operation is not implemented, reading the tag and writing a new value into it are performed as two separate operations. Between the two operations, thread ι' can write its identity into the tag of the target disk on thread ι, and consequently, thread ι and ι' move the same disk. The atomic compare-and-swap operation solves this problem by allowing no operation between the read and write. A detailed discussion on the compare-and-swap operation can be found in Remark 9 in the attached publication 1 [71].

The C++ compiler sometimes rearranges operations for better performance. However, the position and the local time of an active disk must be updated before the disk is released and becomes static. This is guaranteed by the implementation of the memory order `memory_order_seq_cst` in the atomic compare-and-swap operation, which forbids moving operations across the CAS. Furthermore, when a thread acquires a disk, it checks whether the position of the disk is unchanged since the time-of-flight computation, in order to confirm that the disk has not been moved and then released by other threads before being acquired. The CAS, together with the check, serves as an effective lock, implemented in place of a true lock which blocks the code accessing the local time and position and slows down the program. This is referred to as lock-free programming, discussed in Remark 10 and Remark 12 in the attached publication 1 [71]. As only the position is checked in the effective lock, the order of updating the position and local time is crucial. This will be proven below by formal verification.

The Algorithm 1 is complicated, and it is difficult to answer the following questions: if the program finishes without any violation of horizon conditions, is the final configuration always the same? Is there any deadlock in the program? Also, we are interested in knowing whether changing the order of updating the position and the local time impacts the program. All of these questions can be answered by sequential consistency of multithreaded ECMC. We study two-thread runs in a four-disk and a five-disk system for a short chain length by mapping them to absorbing Markov chains. We process the two-thread run using a single-thread program simulating a multithreaded processor. The operation to be executed is referred to as a buffer. For the two-thread case, the buffer is expressed as a pair of

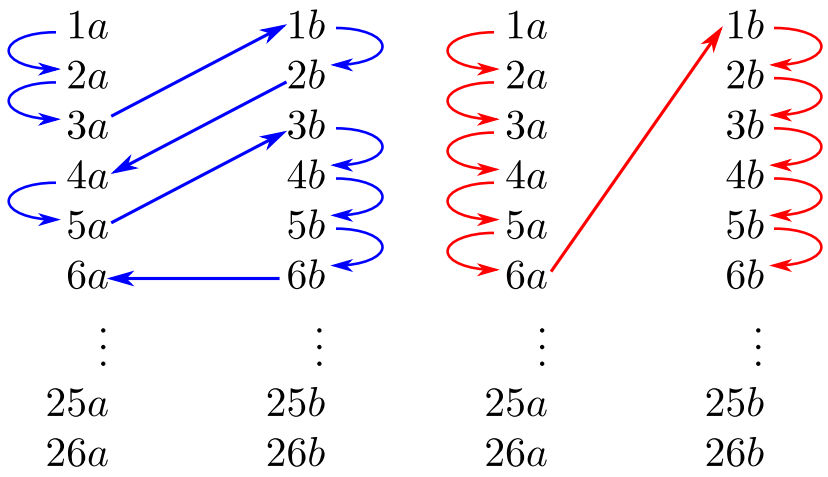

Fig. 5.1 *Examples of orders of executions for a two-thread run. Both threads can either run at the same pace or totally be de-synchronized with each other. The probability of observing a specific order that leads to the wrong outcome (if it exist) may be vanishing.*

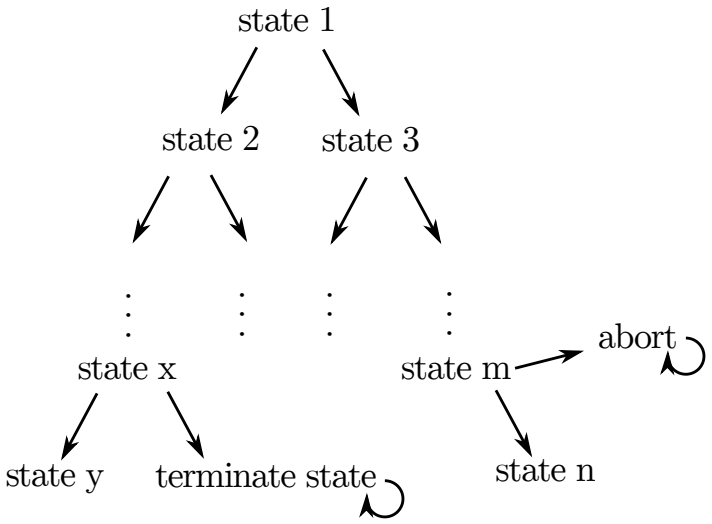

Fig. 5.2 *How the 3670 states are created. Starting from the initial state, two choices of threads generate two states. All the unique states are stored to generate new states. This process is repeated until all the states have been discovered.*

operation indices. The state of the program, which allows for describing how the program runs, contains the disk objects, all local variables, and the buffer. The state contains all the information of the program, in the sense that executing an operation on a given thread evolves a state to another in a deterministic manner. There is a unique state "abort", indicating that the program is terminated due to the violation of horizon condition. The "abort" state does not have buffer content. The simulated multithreaded processor randomly selects one of the two threads and processes the buffer content on the selected thread. In a single run, the operations can be executed following a large number of possible orders (as for example in Fig. 5.1).

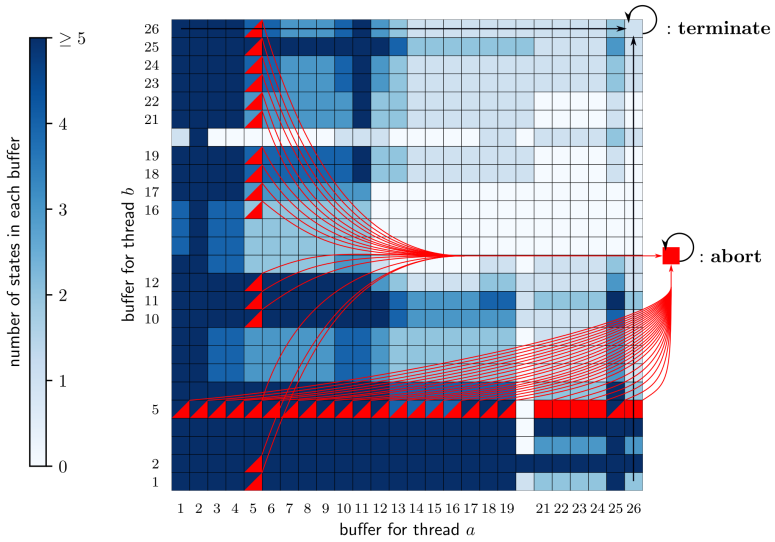

Fig. 5.3 *The* 3670 *states in Alg. 1 for a 4-disk system projected onto the buffer. The* **terminate** *buffer* $\{26a, 26b\}$ *corresponds to a single state.* (*Figure from* [71].)

We exhaust all possible orders and find all possible states. Denoting the threads by a and b, when the simulated multithreaded processor makes a move, the state of the program either transits to the state obtained by executing an operation on thread a or thread b. When the violation of the horizon condition is detected, the state leading to horizon violation transits to the state "abort". For each discovered state, we find the two possible states obtained by executing the buffer content on either of the threads. All the discovered states are stored in a set. We repeat this process until all the new states are in the set of explored states. This procedure is demonstrated in Fig. 5.2. There are 3670 states in the 4-disk case we study. A buffer can correspond to multiple states. The "terminate" buffers $\{26a, 26b\}$ have only one state, indicating that the lifted configuration when the program terminates without any horizon violation is always the same. All states that are not "abort" or "terminate" die out at large times, indicating that there is no deadlock in the program. All states may be projected onto their buffer and visualized (see Fig. 5.3). When applying the same analysis to a 5-disk configuration, exchanging operation 15ι and 16ι in Alg. 1 leads to two terminate states. This result indicates that the order of operation 15ι and 16ι is crucial, as exchanging them can lead to a wrong final configuration. The detailed discussion related to sequential consistency of the multithreaded ECMC can be found in Section 2.3 and 2.4 in the attached publication 1 [71].

This case study of multithreaded ECMC demonstrates the strength of formal verification. When running the C++ program, exchanging operations 15 and 16 produce virtually no wrong output. Also, the rare wrong output is not reproducible, as the randomness in the multithreaded processor cannot be seeded. Running the program repeatedly and not seeing undesired output does not prove the program is correct. Yet the reordering problem is detected by formal verification.

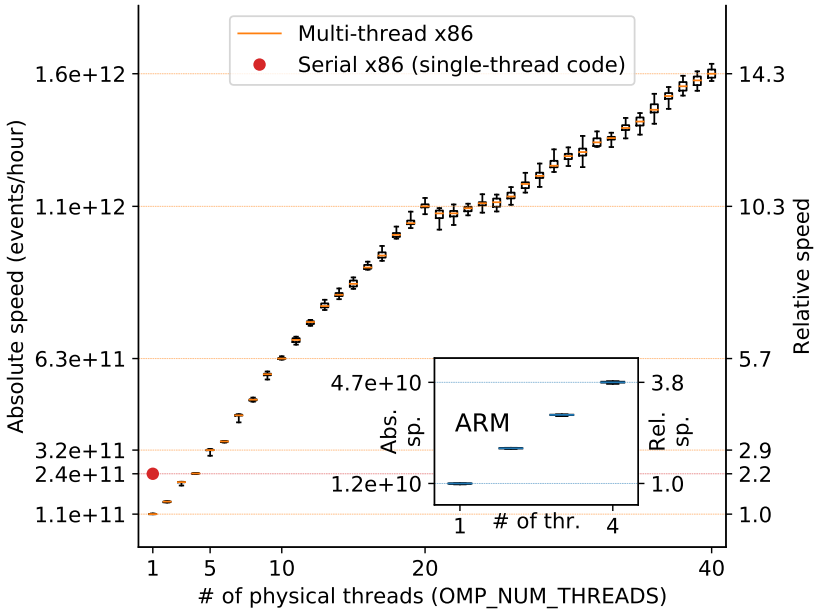

Fig. 5.4 *The EPH for multithreaded ECMC between two breakpoints for a 20 cores (40 threads) Intel x86 CPU and an ARM CPU. The number of active disks is always* 80*. The single-thread code implements the pre-computed constraint graph but no atomic variable. Compared to the cell-based version, the constraint graph introduces a speedup of* 10*. The speed on the x86 CPU increases linearly for 1 to 20 threads and 21 to 40 threads with two different slopes. At 40 threads, the program reaches* 1.6×10^{12} *EPH. On an ARM CPU, the absolute speed is ten times slower. However, the performance scale perfectly with respect to the number of threads and is better than the x86 CPU. This may be due to the various frequency of the x86 CPU. (Figure from* [71]*.)*

5.2 Performance of computer programs

In chapter 3, we discussed the performance measured in the number of moves. However, depending on the implementation, the computer time it takes to process a move differs drastically. We measure the performance of the computer programs by the number of moves (events) performed per hour (EPH).

The multithreaded ECMC has multiple implementations, all in C++, using OpenMP, benchmarked on Intel Xeon 6230, 20 cores (40 threads) @ 2.10(3.90)GHz. The implementation discussed in section 3.3 in the attached publication 1 [71], whose performance is shown in Fig. 5.4, treats only the run between two breakpoints. The constraint graph is precalculated, and the horizon violation is ignored. This implementation manages to achieve 1.6×10^{12} EPH, 100 times the performance of the serial implementation. Combining the EPH with the performance discussed in chapter 3, we arrive at the conclusion that the $|\Psi_6|$ can decouple with the extreme initial condition in hours. With this implementation, we show that ECMC algorithms can be parallelized, even though this was finalized only partially, as the horizon violations have to be taken care of for practical use. Our work-in-preparation implementation of multithreaded ECMC is able to back up the lifted configuration at each breakpoint, and rewind to the last breakpoint if a horizon violation is detected. When the

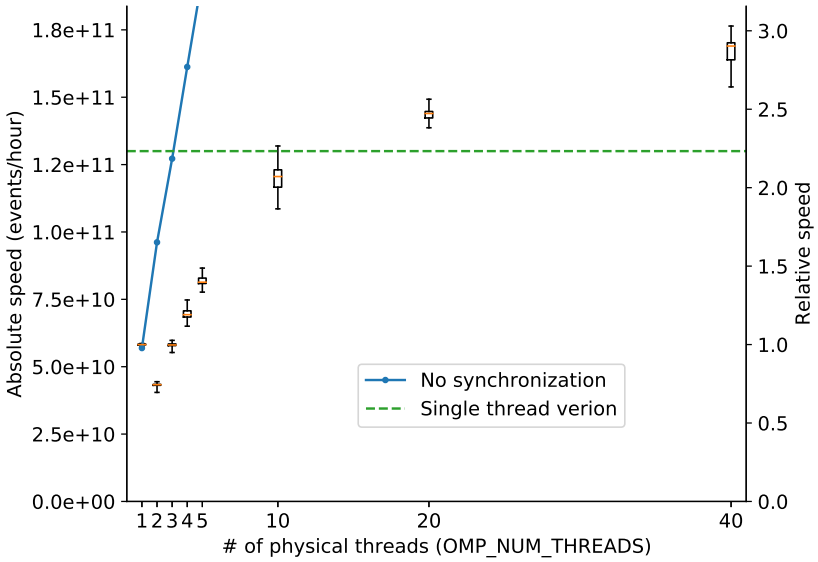

Fig. 5.5 *The EPH for multithreaded ECMC between two resamplings in an implementation that rewinds the system to the last breakpoint in the case of a horizon violation, measured on the same x86 CPU as in Fig. 5.4. The blue curve is the reference between-breakpoint performance. This curve is lower than the curve in Fig. 5.4 due to the overhead related to rewinding. The impact of the horizon violation manifests as the two-thread performance being worse than the single-thread performance. The performance scales linearly with respect to the number of threads until 10 threads, then the return of adding more threads becomes diminishing. At 40 threads the program reaches* 1.8×10^{11} *EPH, roughly ten times slower than the implementation in Fig. 5.4.*

horizon violation takes place, the program has to restart from the last breakpoint, and some of the computation has to be repeated. Preparing and recovering from backup are also time-consuming. As a consequence, as shown in Fig. 5.5, this implementation is only 10 times faster than the serial implementation.

However, as the direction of velocity is required to change frequently, the constraint graph has to be computed frequently. The third implementation of multithreaded ECMC takes into account the computation of constraint graphs, which introduces a huge bottleneck and bring the performance down to $\leq 10^{11}$ EPH. Again, this is also a work in preparation. In the third implementation, the constraint graph is computed on the CPU. In the constraint graph computation, the hard-disk configuration is read-only, and each memory location in the constraint graph is written to only once, allowing for massive parallelization. We have attempted the massively parallel constraint-graph computation on the GPU. Still, the performance is limited by the bandwidth of transmitting data between the GPU and CPU memory. We are waiting for further hardware that may solve the bandwidth problem. Nevertheless, we have shown that the full implementation of a parallelized ECMC is possible.

The state-of-the-art single-thread program for hard disks is straight ECMC implemented in C++. The number of events per hour is roughly 10^{10}. This means evolving the $|\Psi_6|$ of an ordered or a disordered configuration to its typical value takes roughly several days. This implementation of ECMC features a cell system. The constraint graph is not implemented, as the direction has to be changed frequently, and the constraint graph needs to be recalculated [57]. Implementing the constraint graph can speed up ECMC for ten times, but the computer time consumed by the serial constraint-graph computation is huge.

By the time when this thesis is written, the best-performing program is the GPU implementation of MPMC in C++. On a single NVIDIA GeForce RTX3090 GPU, the MPMC code reaches 2.1×10^{13} moves per hour [70], indicating that evolving the $|\Psi_6|$ of a completely ordered (or completely disordered) initial configuration to its typical value takes roughly two days. The performance of the MPMC program is an order of magnitude better than an earlier implementation [29, Table II], and it is likely to further increase. Compared to the Metropolis algorithm, the massive parallelization has a price to pay. The disks are confined in the checkerboard cells in MPMC, and the checkerboard needs to be repositioned. Repositioning the checkerboard is computationally cheap. However, it still make the each move in MPMC slightly less efficient than in the Metropolis algorithm.

Now, it would be interesting to see whether other variants of ECMC (and of event-driven algorithms in general) can be parallelized as was described in the present chapter. One of the main limitations of our approach is the reliance on the constraint graph (which appears restricted to the straight variant of ECMC). Clearly, much work is required to understand whether ECMC algorithms can be successfully parallelized.

Chapter 6

Statistical analysis

In the example of the hard-disk problem, that is the subject of this thesis, the computer outputs a correlated sequence of hard-disk configurations, that is, of samples or of observables such as the pressure. To simplify, we consider only observables that can be expressed as functions of a single sample. Statistical analysis of the sequence guarantees that the running averages roughly correspond to the expectations (as guaranteed in the infinite-time limit) by the ergodic theorem for Markov chain, and that the uncertainty is correctly treated. As for any Markov chain, the computer outputs correlated data. Although samples spaced by more than the correlation time are practically independent, it is usually not optimal to analyze only decimated data (for which the most naive methods that assume the sample to be independent can be used). In this chapter, we introduce the statistical analysis used throughout this thesis.

In the present chapter, we introduce to the statistical analysis in the hard-disk simulation, especially pressure computation, as well as the statistical tools required in the analysis. We verify our implementation of the sampling algorithm by comparing the disk-position distribution, which can be done by performing a variant of the Kolmogorov–Smirnov test [119]. We are also interested in the error bar of pressure, which is given either by blocking [35, 53] or stationary bootstrap [94, 97, 100, 101]. The content in this chapter is only alluded to in the Appendix A.2 in the attached publication 3 [70]. It was however used in all our publications but not treated in detail. The analysis discussed in the present chapter applies to mixed Markov chains. As will be discussed in chapter 7, in realistic settings, the length of the run is usually shorter than the mixing time. Reliable analysis relies on not only rigorous methods discussed in the present chapter but also studying special observables as discussed in chapter 7.

6.1 Distribution comparison

As an example of inference problems typically appearing in this thesis, we want to test whether two implementations of the reflective ECMC algorithms, one with resampling and the other without resampling, sample the same distribution. These implementations can be compared by examining the difference between their spatial density. As discussed in chapter 4, the comparison is usually done on the level of observables (that is, on a statistic of the samples, rather than the configurations themselves). A convenient statistic is the

distribution of one-dimensional projections of disk positions, represented by the empirical distribution (4.18), which is rewritten as

$$\hat{F}_n(x) = \frac{\sum_{i=1}^{n} I(X_i \le x)}{n}, \tag{6.1}$$

where

$$I(X_i \le x) = \begin{cases} 1, & \text{if } X_i \le x; \\ 0, & \text{else.} \end{cases} \tag{6.2}$$

There X_i are the sampled x positions, and $n = Nn_{\text{sample}}$ is their total number. Let the cumulative distribution function be $F(x)$, the Dvoretzky-Kiefer-Wolfowitz inequality for independent samples,

$$P\left(\sqrt{n}\sup_x |F(x) - \hat{F}_n(x)| > \epsilon\right) \le \mathrm{e}^{-2\epsilon^2}, \tag{6.3}$$

suggests that the empirical distribution converges to the cumulative distribution, and the statistic $\sup_x |F(x) - \hat{F}_n(x)|$ scales as $n^{-\frac{1}{2}}$ [27, 79, 119]. As the positions of the disks may be correlated, the bound in (6.3) may no longer be true for the hard-disk position distribution. Nevertheless, the $n^{-\frac{1}{2}}$ scaling remains.

The DKW inequality and the scaling of the statistic $\sup_x |F(x) - \hat{F}_n(x)|$ provide a simplified example of what happens in reality. In practice, the true distribution is not known, and two empirical distributions are compared with each other. The difference of two empirical distributions $\hat{F}_n(x)$ and $\hat{G}_m(x)$ is characterized by the statistics defined as

$$D_{n,m} = \sup_x |\hat{F}_n(x) - \hat{G}_m(x)|. \tag{6.4}$$

In the limit of $n \to \infty$ and $m \to \infty$, if $F = G$, the statistics $D_{n,m}$ satisfies [43]

$$P\left(\sqrt{\frac{mn}{m+n}} D_{n,m} \ge \epsilon\right) \to 2\left(\mathrm{e}^{-2\epsilon^2} - \mathrm{e}^{-2(2\epsilon)^2} + \mathrm{e}^{-2(3\epsilon)^2} - \ldots\right). \tag{6.5}$$

Assuming that we use only a fraction $k < 1$ of all the samples to construct the empirical distributions such that $n = kn'$, $m = km'$, where the total number of samples are n' and m'. Equation (6.5) becomes

$$P\left(\sqrt{k\frac{m'n'}{m'+n'}} D_{k,n',m'} \ge \epsilon\right) \to 2\left(\mathrm{e}^{-2\epsilon^2} - \mathrm{e}^{-2(2\epsilon)^2} + \mathrm{e}^{-2(3\epsilon)^2} - \ldots\right). \tag{6.6}$$

In the limit $kn' \to \infty$ and $km' \to \infty$, the statistics $D_{k,n',m'}$ scales as $k^{-\frac{1}{2}}$. In practice, we prepare two big samples of size n' and m', respectively. Then we chose a fraction k out of both samples, and calculate the statistic $D_{k,n',m'}$. If both samples are statistically the same, the statistics $D_{k,n',m'}$ scales as $k^{-\frac{1}{2}}$. If the samples are obtained from different distributions, there will be persisting deviation from zero in $D_{k,n',m'}$. Practically, we confirms that two distributions are identical if the scaling $k^{-\frac{1}{2}}$ is observed and $D_{1,n',m'}$ becomes sufficiently small (typically 10^{-4} for the density distribution of (6.1)). We use this method to validate implementations of sampling algorithms, and also to study the necessity of resampling in ECMC algorithms. This method assumes that configurations are sampled from the stationary distribution. The scaling is not observed if the Markov chain is not mixed.

6.2 Confidence interval of correlated sequences

To continue with the description of the hard-disk model as an example of a general estimation problem, the pressure is the expectation of its estimator. To characterize the fluctuation of this value, we calculate its confidence interval, that is, to find an interval $(\hat{P}_{\alpha,\text{low}}, \hat{P}_{\alpha,\text{high}})$ such that, for a probability larger than $1-\alpha$, the expectation of pressure falls inside this interval [119]. If the distribution of the pressure estimator is symmetric, the confidence interval is centered at the expectation. The half-length of a confidence interval is usually referred to as the "error bar" for $\alpha \approx 0.32$.

In order to get hold of the fluctuation of the pressure estimator $\hat{P}$, the whole run is broken into n_{sequence} consecutive small runs, and the pressure estimator is evaluated in each of them. The pressure is given as a correlated time series $\hat{P}_1, \hat{P}_2, ..., \hat{P}_{n_{\text{sequence}}}$. The value of the estimator during the whole run is the average of the time series, $\hat{P} = \sum_{t=1}^{n_{\text{sequence}}} \hat{P}_t / n_{\text{sequence}}$. For independent time series, the error bar of its average is the standard error $\sigma/\sqrt{n_{\text{sequence}}}$ of the time series, where

$$\sigma = \sqrt{\frac{\sum_{t=1}^{n_{\text{sequence}}} (\hat{P}_t - \hat{P})^2}{n_{\text{sequence}} - 1}} \tag{6.7}$$

is the standard deviation of the time series. In pressure computation, there are usually correlations in the time series that complicates the problem. The correlation is characterized by the auto-correlation function $C(\tau)$ defined as

$$C(\tau) = \frac{\sum_{t=1}^{n_{\text{sequence}}-\tau} (\hat{P}_t - \hat{P})(\hat{P}_{t+\tau} - \hat{P})}{(n_{\text{sequence}} - \tau)\sigma^2}. \tag{6.8}$$

Usually the auto-correlation function $C(\tau)$ approaches 0 as τ approaches ∞. It defines a time scale referred to at the auto-correlation time, beyond which two elements in the time series are practically independent. For a correlated time series, the standard deviation is larger than (6.7), since (6.7) treats every element in the series as independent, thus over-estimated the information carried by each element. The error bar of a correlated time series is obtained mainly by two methods: blocking and stationary bootstrap. The error bar is also obtained by doing independent runs and finding the standard error of their results. However, these runs usually starts from correlated (if not identical) initial configurations, and the correlation of these "independent" runs is not well controlled. Thus, obtaining the results in a long run and presenting the result in a consistently correlated time series is preferred over "independent" runs, unless the independence of the runs has been thoroughly studied.

Inspired by the renormalization group theory [122], the blocking method is first proposed in [35] to analyze correlated time series. In the blocking method, the operation of averaging two pressures next to each other is applied to the time series repeatedly. For example, the time series $\hat{P}_1, \hat{P}_2, ..., \hat{P}_{n_{\text{sequence}}}$ becomes $\hat{P}'_1, \hat{P}'_2, ..., \hat{P}'_{n_{\text{sequence}}/2}$, where $\hat{P}'_i = (\hat{P}_{2i-1} + \hat{P}_{2i})/2$. Each operation reduces the length of the time series by half. The number of original elements, namely $\hat{P}_1, ..., \hat{P}_{n_{\text{sequence}}}$, in each transformed element is thus $1, 2, 4, 8, ...$ as the operation is applied repeatedly to the time series. The transformed element can be seen as a sub-series. The auto-correlation function of the transformed elements is calculated using (6.8) after substituting the original elements in the formula into transformed elements. So, if the length of the sub-series, referred to as the block size, is much larger than the auto-correlation time, the auto-correlation function of the transformed time series is zero except

for at $\tau = 0$. In this case, the standard error of the transformed time series is the error bar, and further transform keeps it unchanged. The computed standard error has its own uncertainty. If the block size becomes too large compared to the length of the original series, the transformed series has too few elements. The uncertainty of the standard error becomes too large, resulting in large fluctuation in the computed error bar. In practice, the blocking is performed by plotting the standard error, together with the uncertainty of the standard error, as a function of the block size. The standard error increases with respect to the block size at the beginning. Then, a plateau emerges. At last, when the block size is comparable with the length of the series, the standard error of the transformed series can take any value due to the fluctuation. The error bar of the time series is the value of standard error at the plateau. If the fluctuation becomes significant before the plateau showing up, the time series is too short for statistical analysis. The optimal block size can be obtained automatically [53].

Another method of calculating the error bar is the stationary bootstrap [100]. Similar to the idea in the bootstrap [28], the stationary bootstrap duplicates the time series as pseudo bootstrap time series $\{\hat{P}_t^b, t = 1, ..., n_{\text{sequence}}\}, b = 1, ..., n_b$ by shuffling. Denote the average of the pseudo bootstrap time series as $\hat{P}^b$, the standard deviation of the average of the bootstrap time series

$$\sigma_b = \sqrt{\frac{\sum_{b=1}^{n_b} \left(\hat{P}^b - \frac{1}{n_b}\sum_{b=1}^{n_b} \hat{P}^b\right)^2}{n_b - 1}} \tag{6.9}$$

is the error bar of the original time series. As it is possible to generate an infinite number of the bootstrap time series, the uncertainty of σ_b can be arbitrarily small. Shuffling in the stationary bootstrap takes into account the correlation between the elements. There is a parameter $p \in (0, 1)$ in the stationary bootstrap, namely the probability of choosing a random element in the original time series when adding an element into the bootstrap time series. The bootstrap time series is created starting from choosing a random element in the original time series and adding it to the empty bootstrap time series. Denoting the last element added to the bootstrap time series as $\hat{P}_t$, with probability $1 - p$, $\hat{P}_{t+1}$ is added into the bootstrap time series. $\hat{P}_{n_{\text{sequence}}+1}$ is regarded as $\hat{P}_1$. With probability p, a random element in the original time series is added to the bootstrap time series. Repeat this process until the length of the bootstrap time series is equal to the length of the original time series. The value of p is chosen such that the mean square error of the error bar calculated by (6.9) is minimized [94, 97, 100, 101].

The blocking method and stationary bootstrap are the statistical analyses we have performed in the attached publication 3. The cross-validation of pressure estimators in the same publication also relies on the precise computation of error bars. The error bars constitute an interval estimator, while the running average of the pressure is a point estimator. Both of the methods we present in this section yield reliable confidence interval for mixed Markov chains. However, as will be discussed in chapter 7, these methods along can underestimate error bars and lead to biased results if the Markov chain is not mixed well.

Chapter 7

Interpreting simulations

Interpreting the output of computer programs is the uppermost step in the stair-case structure in Fig. 1.1, where conclusions in physics are drawn from the numbers obtained from an algorithm (step 1), that correspond to a certain observable (step 2), implemented in software (step 3), that are validated (step 3), and that have reliable error bars (step 4). For example, the numerical equation of state, that is, the computed pressure as a function of volume, has once served as an evidence of the phase transition. However, in the work of this thesis, we focus on the method of hard-disk simulation instead of the physics, and we are still using the conceptual framework of the phase transition scenario proposed in [10] and confirmed in [29]. We are interested in the information encoded in the numerical outputs about computation itself. In this chapter, we discuss the information in the output that concerns the convergence of the computation and the reliability of results in the case of non-convergence. The present chapter discusses again the statistical analysis. However, in this chapter, the analysis is performed after, and based on the statistical analysis discussed in chapter 6, as it is more about the implication of the numerical results.

In this chapter, we discuss the statistics that characterize the ergodicity and the non-equilibrium behavior of a Markov chain. We present two well-monitored example runs (shown in Fig. 7.1 and Fig. 7.2) using state-of-the-art programs taking place in a $N = 870$ and a $N = 128^2$ system at density $\eta = 0.716$, at which the majority of the system is hexatic. Based on our observations in these runs, we demonstrate that: the global orientational order is able to be used as a diagnostic tool for the convergence of Markov chains; the convergence of pressure and the "rotation" of global orientational order are connected to each other as they are statistically correlated in small systems; the long-lasting fluctuation in pressure may lead to biased computation. Also, we show that even today, for large systems, convergence in the strict sense of that the probability distribution at large times is independent of the initial configuration is impossible. But the pressure computation in these systems remains valid, as the pressure is virtually independent of the global orientational order. In the attached publication 3, we perform hard-disk simulations for parameters that were previously considered in the literature, show the synopsis of published results and their modern reproduction, and interpret the possible difference between them. We also provide our precise computation of pressures in a wide range of systems as a benchmark for future works.

7.1 Functions of pressure and global orientational order

In this section, we introduce to the functions of pressure and orientational order that are crucial for the statistical validity of computation. Unlike our discussion in chapter 6, we discuss the meaning of these functions in stochastic computation rather than how the analysis is performed. The relevant discussion can be also found in section IV in the attached publication 3 [70].

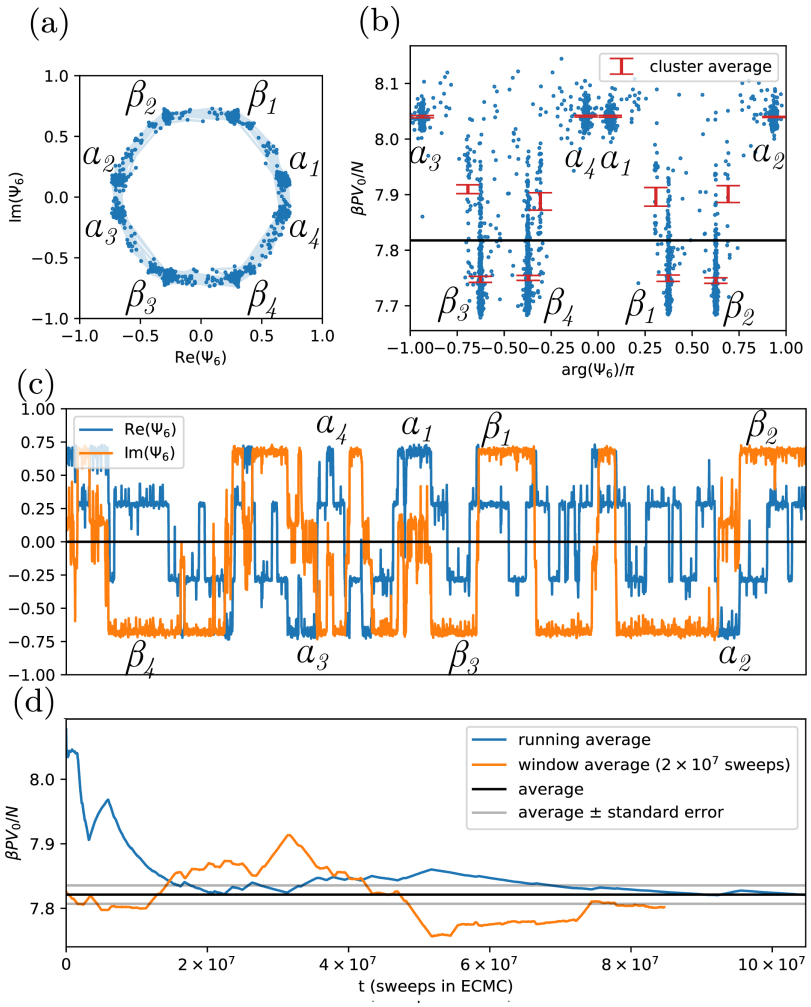

Fig. 7.1 *Pressure P and global orientational order Ψ_6 for a three-hour ECMC run ($N = 870$, $\eta = 0.716$, $\alpha = (1:1)$). (a): Values of Ψ_6 in the complex plane. Highlighted clusters with inverted Ψ_6 (such as α_1 and α_3) have the same statistical weight. (b): Cluster averages for P vs. $\arg(\Psi_6)$. (c): Trajectories of $Im(\Psi_6)$ and $Re(\Psi_6)$ with indicated clusters. (d): Running average and window average for P. (Figure from [70].)*

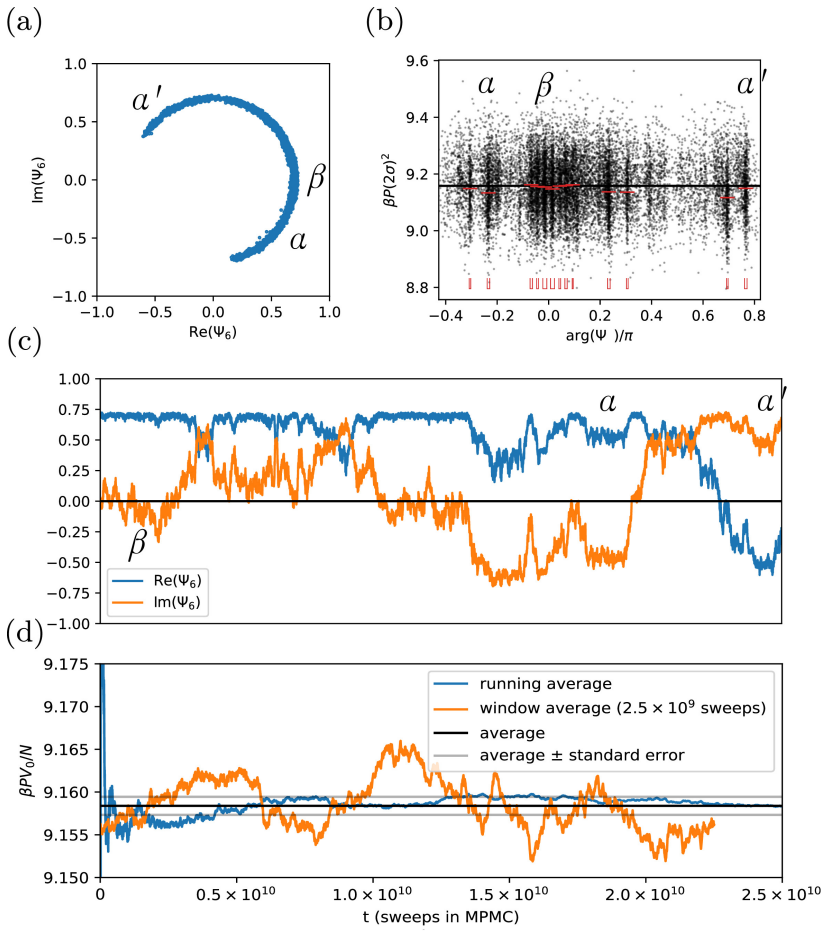

Fig. 7.2 *Pressure P and global orientational order Ψ_6 for a single MPMC run ($N = 128^2$, $\eta = 0.716$, $\alpha = (1:1)$). (a): Histogram of Ψ_6 in the complex plane. Clusters α, α' satisfies $\Psi_6^{\alpha} = -\Psi_6^{\alpha'}$ and have equal weight. (b): Cluster averages for P vs. $\arg(\Psi_6)$. (c): Trajectory of $Re(\Psi_6)$ and $Im(\Psi_6)$ with first visits to clusters indicated (cf Fig. 7.1). (d): Running average and window average of the pressure, showcasing slow convergence. (Figure from [70].)*

7.1.1 The sequence of global orientational order Ψ_6

As discussed in section 4.4, the global orientation has symmetry, and its expectation in a square box is known to vanish. Suppose that the global orientational order is computed for configurations sampled at discrete times labeled by $t = t_0, t_1, ..., t_{n_{\text{sample}}-1}$, the ergodic theorem of Markov chain guarantees that the running average of the sequence $\{\Psi_6(\mathbf{x}(t)), t_0, t_1, ..., t_{n_{\text{sample}}-1}\}$ approaches 0 for $n_{\text{sample}} \to \infty$. In practice this is realized in two steps. In the first step, the $|\Psi_6(\mathbf{x}_t)|$ decouples from the initial condition $|\Psi_6(\mathbf{x}_0)|$, as demonstrated in chapter 3. In the second step, the $\Psi_6(\mathbf{x}(t))$ stays in the ring of the typical $|\Psi_6|$ in equilibrium, and explores the space of $\arg\Psi_6$.[1] In small systems, the exploration features jumps

[1] Nothing prevents the two steps takes place together. However, for high enough densities, in the hexatic phase, the convergence of $|\Psi_6|$ always happens before the full exploration of the space of $\arg\Psi_6$.

from one side to the other side on the ring and prolonged exploration in a cluster, as shown in Fig. 7.1(a)(c). For large systems, the exploration features fast local oscillations and slow global drifts, as shown in Fig. 7.2(a)(c). If the run if long enough for the ergodic theorem to manifest, the whole ring is explored in a symmetric manner, as shown in Fig. 7.1(a). However, in practical pressure computations, the system is too large such that days of computer time still leave an opening in the ring, as shown in Fig. 7.2(a) for a $N = 128^2$ system. It can be even worse, for example, for a even larger system of $N = 1024^2$, shown in Fig. 7.3(b), that the $\arg \Psi_6$ remains almost the same throughout the whole run. This is observed for multiple initial $\arg \Psi_6$ almost equally spread within the interval $[0, \pi/2]$. Strictly speaking, if the running average of Ψ_6 does not reach 0 or the ring is not fully explored, the Markov chain is too short to be ergodic. However, the symmetry discussed in chapter 4 indicates that exploring one fourth of the complex plane amounts to sampling configurations having all possible orientation.

Fig. 7.1(a) shows the scattering plot of $\Psi_6(\mathbf{x}(t))$ during a run for a $N = 870$ system at density $\eta = 0.716$ in a square box. The Ψ_6 mainly locates in several clusters which complies with its symmetry. For small systems like this, configurations with specific orientations are more compatible with the box than other configurations. In order words, the boundary free energy, which likely plays an important role in the total free energy, is smaller in these clusters. Fig. 7.2(a)(c) shows the evolution of Ψ_6 in a $N = 128^2$ system at the same density in a square box. The whole space of Ψ_6 is not fully explored, but at least one of the four symmetric regions is well explored. The clusters in the complex plane are less pronounced, as for large system the boundary free energy becomes less significant compared to the volume free energy. For both system sizes the $|\Psi_6|$ is clearly away from zero, as the majority of the system is in hexatic phase. Fig. 7.3(b) demonstrates the scatter plot of pressure versus $\arg \Psi_6$ in a $N = 1024^2$ system in a square box. The Ψ_6 does not rotate at all, indicating that the sample space is not explored well.

The ergodic theorem of Markov chain discussed in chapter 3 states that, in the limit of $n_{\text{sample}} \to \infty$, the running average of $\{\Psi_6(\mathbf{x}(t)), t = t_0, t_1, ..., t_{n_{\text{sample}-1}}\}$ converges to zero.[2] What we find useful in practice is its contraposition: if the running average of Ψ_6 is manifestly away from zero, the Markov chain is not long enough to achieve ergodicity. This proposition leads to a practical criterion we use to check the ergodicity of a run: if the $\arg \Psi_6(\mathbf{x}(t))$ fails to cover a angle more than $\pi/2$, the run is not ergodic. This criterion has already been implemented in previous works [10, 11]. For a non-ergodic run, the running average of all other observables may not converge to their expectation. As we will shortly discuss, the non-convergence of pressure is observed in this case.

7.1.2 Correlation between pressure and global orientational order

The correlation between the pressure and $\arg(\Psi_6)$ is shown in Fig. 7.1(b), Fig. 7.2(b), and Fig. 7.3(a)(b) for $N = 870, 128^2, 512^2, 1024^2$, respectively. For $N = 870$, the correlation between the pressure and $\arg(\Psi_6)$ is clearly observed, as there are clusters in the scatter plot, and the difference of pressure in each of them is much larger than the fluctuation of pressure

[2] This statement has to be taken with a grain of salt because it is strictly valid only for irreducible Markov chains. The Böröczky packing blocks the ECMC algorithms, and none of the variants is strictly irreducible. However, as discussed in chapter 2, Böröczky packings are assumed to play no role.

in each cluster.[3] The clusters in Fig. 7.1(b) correspond to clusters in Fig. 7.1(a). If the Ψ_6 is trapped within one of the clusters in the complex plane, the result of pressure is the biased cluster pressure. In this case, a non-rotating Ψ_6 indicates explicitly a biased pressure.

For the $N = 128^2$ system, there are still clusters in the scatter plot Fig. 7.2(b). However, the difference between the pressure in various clusters becomes much smaller. The dependence is still detectable as the pressure of some clusters clearly differ from that of other clusters, but the relative difference is quantitatively smaller than in the $N = 870$ case. As shown in the Fig. 7.3(a), there is no visible dependence between the pressure and the $\arg \Psi_6$. Fig. 7.3(b) shows that, in a $N = 1024^2$ system, which is the largest system in which pressure is calculated precisely by the time this thesis is written, the systematic deviation in the clusters is comparable with the fluctuation of pressure in each of the clusters. Thus, although no algorithm can rotate Ψ_6 in a $N = 1024^2$ system at the phase-transition density, we can still trust their pressure computation.

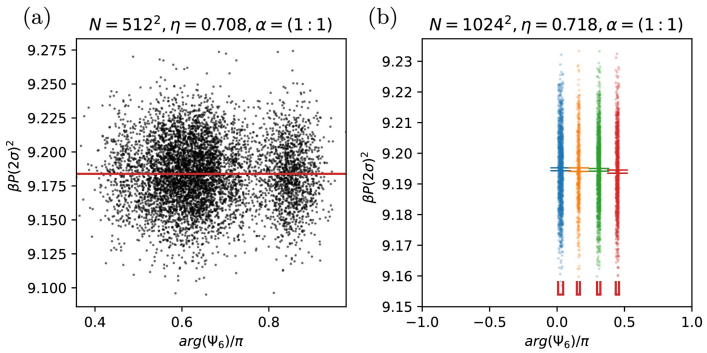

Fig. 7.3 *Convergence analysis for the hard-disk model at* $\eta = 0.708$ *and* $\eta = 0.718$ *(square box* $\alpha = (1:1)$*) for MPMC. (a): Scatter plot of the pressure as a function of the orientational order parameter* $(N = 512^2)$*. (b) Cluster averages obtained from independent runs from initial configurations at specific values of* Ψ_6*.* $(N = 1024^2)$ *The difference, smaller than* 10^{-3}*, estimates the systematic error. The rotation of* Ψ_6 *is not relevant for large systems. (Figure from* [70]*.).*

7.1.3 Window average of pressure

The window average refers to a sequence of length $n_{\text{sequence}} - t_w$, where t_w is the window size, derived from a sequence of pressure $\{\hat{P}_t, t = 1, 2, ..., n_{\text{sequence}}\}$. The window average is calculated as

$$\hat{P}_t^{t_w} = \frac{\sum_{t'=t}^{t+t_w-1} \hat{P}'_t}{t_w}, t = 1, 2, ..., n_{\text{sequence}} - t_w - 1. \tag{7.1}$$

The window average is calculated to average out the short-term fluctuation in the sequence of pressure, and is plotted in Fig. 7.1(d) and Fig. 7.2(d). For $N = 870$ in straight ECMC, there are two period in which the window average of pressure is constantly out of its confidence interval, each lasting for 2×10^7 sweeps in straight ECMC, corresponding to thirty minutes of computer time. If the total run time is shorter than thirty minutes, the pressure can be

[3] The clusters subject to symmetry are considered as one cluster.

biased. This bias can be detected by tracking the $\arg \Psi_6$ in the same run, but is impossible to be detected by studying only the sequence of pressure. For $N = 128^2$ in MPMC, there are periods of significant deviation lasting for roughly 0.5×10^{10} sweeps, corresponding to more than a day. As the MPMC is faster than the Metropolis algorithm by a factor of thousands, the deviated interval corresponds to ten years in the Metropolis algorithm, which indicates that having non-biased results using Metropolis algorithm in $N = 128^2$ systems is very hard. Nevertheless, comparing Fig. 7.2(d) and Fig. 7.2(c) one again concludes that the deviation is correlated to a non-rotating Ψ_6, and this deviation is detectable if Ψ_6 is computed together with the pressure.

7.2 Previous results

To highlight the limitation in the legacy pressure computations, we compare them to our computations done by modern tools. For small systems, the limitation can be the dependence of the aspect ratio, and for large systems, this can be the large mixing time. For both small and large systems, the computation may suffer from non-convergence due to limited computational resource, as well as inefficient pressure estimators using legacy methods presented in chapter 4.

The hard-disk pressure computation is pioneered by Metropolis et al. [84] and Alder and Wainwright [2]. Their results were published more than sixty years ago, when the computational resource was strictly limited. It was impossible to reach equilibrium, as the mixing time was not manageable on the computers back then. Thus, the pioneering works are considered as "computer experiments". They were supposed to provide insights, and the precision of pressure computation was never their priority. However, with the computational resource available to us today, we can obtain reliable numerical pressure in the systems considered in the pioneering works. The synopsis of the historic result and their modern reproduction is shown in Fig. 7.5(a). The Metropolis et al.'s result is systematically lower than the modern results. According to Metropolis et al.'s description of their runs, they start from an approximately crystalline configuration and last at most for eighty cycles. In each cycle, each disk is moved at least once. The Ψ_6 of a reproduced run using Metropolis et al.'s setup and the Metropolis algorithm, implemented (without a cell system) in Python and running in Pypy, is shown in Fig. 7.4. Metropolis et al.'s run finishes in less than twenty seconds on a modern laptop. The crystalline order in the initial configuration is relaxed. However, the space of orientational order is not explored. As in small systems the pressure and orientational order are strongly correlation, the pressure of the computation is likely to be biased. As the runs are short, the simulation of Metropolis et al. is similar to the perturbation of a crystal. The pressure in Metropolis et al. is obtained by extrapolation, and its systematic bias is not known. The Ψ_6 of the reproduction of Metropolis et al.'s work demonstrate the strength of using it as a diagnostic tool. Besides, Metropolis et al. has been scanning a overly large density interval. As a consequence, the resolution of the equation of state is not good enough to identify the Mayer-Wood loop indicating the phase transition.

For the result of Alder and Wainwright [2] shown in Fig. 7.5(b)(c), we first notice that the equation of state depends qualitatively on the aspect ratio. For $N = 72$, the loop indicating the phase transition appears for $\alpha = (9 : 8\sqrt{3}/2)$, but not in a square box. We experiment with the aspect ratio and suppose that the original computation of Alder and Wainwright is

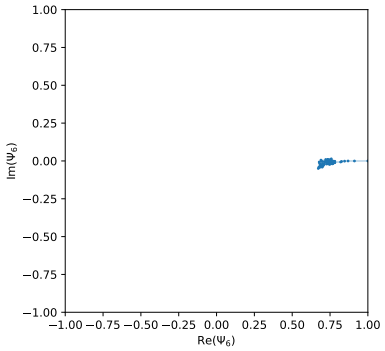

Fig. 7.4 *Ψ_6 for a $N = 224$ system in a square box at density $\eta = 0.708$ lasting for 80 cycles, starting from an approximate triangular lattice. The Ψ_6 is plotted after every cycle. The computation in [84] is reproduced using the same parameters and the length of the run in a modern Python program. $|\Psi_6|$ decreases during the run until reaching a cluster, indicating the order in the initial configuration has relaxed. However, this run does not explore the sample space well, since Ψ_6 does not rotate. The resulting pressure of this run is likely to be biased.*

done with aspect ratios that allow for hosting fully packed configurations in periodic boxes. The result of Alder and Wainwright also suffers from short run time. The first evidence is that the there are two pressure for $N = 72$ as shown in Fig. 7.5(b). We suppose that this is related to having different pressures in clusters corresponds to different Ψ_6 in small systems, as the modern result of pressure is between the two historic ones. For $N = 870$, if the initial configuration is a crystal and the run is short enough, we reproduce the equation of state in [2]. If the Ψ_6 of the run rotates well, the equation of state and Alder and Wainwright's results have a gap in between, as demonstrated in Fig. 7.5. We suppose that some clusters of Ψ_6 correspond to more ordered configurations that have lower pressure, and the system is likely to be trapped in such clusters.

The synopsis of equations of state in $N = 128^2, 512^2, 1024^2$ systems is plotted in Fig. 7.6. In large systems, the aspect ratio becomes irrelevant. Similar to the pioneering works, most of the past works in the medium-large-size systems are "computer experiments". The sampling algorithms used in these works are good enough to generate configurations in large systems, in which some of the finite-size limitations are lifted, and new discussion in physics is made possible. However, they are not good enough to reach equilibrium. This is not a surprise, as our state-of-the-art program is not able to rotate Ψ_6 in a $N = 1024^2$ system even when this thesis is being written. Our computation of five-digit precision confirms the historic results produced by ECMC and MPMC. This motivate us to believe that these results will be confirmed again after decades. We present the cross validation of our results in Table II in the attached publication 3 [70]. However, even the best methods are not able to treat the system larger than $N = 1024^2$ or $\eta > 0.72$, thus, a part of the physics of hard disk still remains unknown. We are still waiting for an algorithm that surpasses all existing methods to fully understand the hard-disk model.

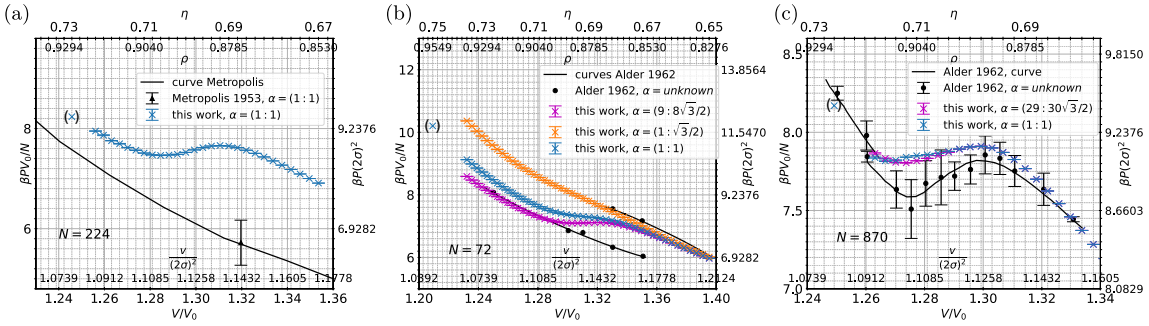

Fig. 7.5 *Equations of state $P(V)$ for the hard-disk model at small N. (a): $P(V)$ for $N = 224$ for $\alpha = (1:1)$ computed in 1953 [84] compared with ECMC computations in [70]. (b): $P(V)$ for $N = 72$ computed in 1962 [2] (for unspecified aspect ratio α) and ECMC pressures for $\alpha = (1:\sqrt{3}/2)$, $(1:1)$, and $(9:8\sqrt{3}/2)$. (c): $P(V)$ for $N = 870$. This work's square-box computations satisfy $\langle \Psi_6 \rangle \simeq 0$, except for data points in parentheses (see Fig. 7.1). (Figure from [70].)*

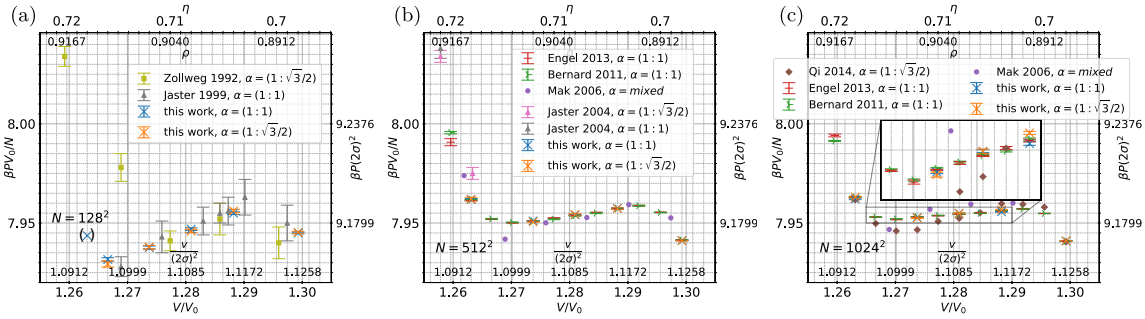

Fig. 7.6 *Equations of state $P(V)$ for the hard-disk model at large N. (a): $P(V)$ for $N = 128^2$ from [50, 70, 135] for $\alpha = (1:\sqrt{3}/2)$ and $\alpha = (1:1)$ where all but the data point in parentheses satisfy the rotation criterion. (b): $P(V)$ for $N = 512^2$ from [29, 52, 70, 78, 78] for aspect ratios $\alpha = (1:\sqrt{3}/2)$ and $\alpha = (1:1)$, where runs with $\eta < 0.712$ satisfy the rotation criterion. (c): $P(V)$ for 1024^2 from [10, 29, 70, 78, 102], for aspect ratios $\alpha = (1:\sqrt{3}/2)$ and $(1:1)$, where at density $\eta > 0.708$ the rotation criterion is violated, but the systematic error thus committed is negligible. (Figure from [70].)*

Bibliography

[1] B. J. Alder & T. E. Wainwright 1959. "Studies in Molecular Dynamics. I. General Method". *J. Chem. Phys.* **31**, pp. 459–466. Cited pages 29 and 39

[2] B. J. Alder & T. E. Wainwright 1962. "Phase Transition in Elastic Disks". *Phys. Rev.* **127**, pp. 359–361. Cited pages 4, 7, 84, 85, and 86

[3] M. P. Allen 2006. "Evaluation of pressure tensor in constant-volume simulations of hard and soft convex bodies". *J. Chem. Phys.* **124**(21), p. 214 103. Cited page 61

[4] Z. Allen-Zhu, Y. Li & Z. Song 2019. "A Convergence Theory for Deep Learning via Over-Parameterization". In "Proceedings of the 36th International Conference on Machine Learning", , edited by Kamalika Chaudhuri & Ruslan Salakhutdinov *Proceedings of Machine Learning Research*, volume 97 (PMLR) pp. 242–252. URL `https://proceedings.mlr.press/v97/allen-zhu19a.html`. Cited page 11

[5] J. A. Anderson, E. Jankowski, T. L. Grubb, M. Engel & S. C. Glotzer 2013. "Massively parallel Monte Carlo for many-particle simulations on GPUs". *J. Comput. Phys.* **254**, pp. 27–38. Cited pages 4, 29, and 34

[6] S. Asakura & F. Oosawa 1954. "On Interaction between Two Bodies Immersed in a Solution of Macromolecules". *J. Chem. Phys.* **22**, pp. 1255–1256. Cited page 20

[7] F. Bailly & G. Longo 2011. *Mathematics and the natural sciences: the physical singularity of life* volume 7 (World Scientific). Cited page 1

[8] Y. Baryshnikov, P. Bubenik & M. Kahle 2013. "Min-Type Morse Theory for Configuration Spaces of Hard Spheres". *Int. Math. Res. Not.* **2014**, pp. 2577–2592. Cited page 25

[9] R. J. Baxter 2016. *Exactly Solved Models in Statistical Mechanics* (Elsevier). Cited page 1

[10] E. P. Bernard & W. Krauth 2011. "Two-Step Melting in Two Dimensions: First-Order Liquid-Hexatic Transition". *Phys. Rev. Lett.* **107**, p. 155 704. Cited pages 7, 31, 35, 63, 79, 82, and 86

[11] E. P. Bernard, W. Krauth & D. B. Wilson 2009. "Event-chain Monte Carlo algorithms for hard-sphere systems". *Phys. Rev. E* **80**, p. 056 704. Cited pages 4, 6, 29, 31, 34, 35, 36, and 82

[12] Etienne Bernard 2011. *Algorithms and applications of the Monte Carlo method: Two-dimensional melting and perfect sampling*. Ph.D. thesis Université Pierre et Marie Curie - Paris VI. URL https://tel.archives-ouvertes.fr/tel-00637330. Cited page 21

[13] D. Bernoulli 1738. *Hydrodynamica*. URL http://dx.doi.org/10.3931/e-rara-3911. Cited pages 3 and 18

[14] H. J. Boehm & L. Crowl 2009. "C++ Atomic Types and Operations ". http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2427.html. Cited page 5

[15] A. Bordes, S. Chopra & J. Weston 2014. "Question answering with subgraph embeddings". In "Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)", (Association for Computational Linguistics, Doha, Qatar) pp. 615–620. URL http://dx.doi.org/10.3115/v1/D14-1067. Cited page 8

[16] K. Böröczky 1964. "Über stabile Kreis- und Kugelsysteme". *Ann. Univ. Sci. Budapest. Eötvös Sect. Math.* **7**, pp. 79–82. Cited pages 6, 22, 23, and 24

[17] L. Bottou 2012. *Stochastic Gradient Descent Tricks Lecture Notes in Computer Science (LNCS)*, volume 7700 (Springer) neural networks, tricks of the trade, reloaded edition pp. 430–445. URL https://www.microsoft.com/en-us/research/publication/stochastic-gradient-tricks/. Cited page 11

[18] E. Brini, C. Simmerling & K. Dill 2020. "Protein storytelling through physics". *Science* **370**(6520). Cited page 2

[19] L. M. Brown & R. T. Weidner. "physics". *Encyclopedia Britannica* . Cited page 1

[20] P. E. Brumby, A. J. Haslam, E. de Miguel & G. Jackson 2011. "Subtleties in the calculation of the pressure and pressure tensor of anisotropic particles from volume-perturbation methods and the apparent asymmetry of the compressive and expansive contributions". *Mol. Phys.* **109**(1), pp. 169–189. Cited page 61

[21] A. Choromanska, Y. LeCun & G. B. Arous 2015. "Open problem: The landscape of the loss surfaces of multilayer networks". In "Conference on Learning Theory", (PMLR) pp. 1756–1760. Cited page 10

[22] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu & P. Kuksa 2011. "Natural language processing (almost) from scratch". *J. Mach. Learn. Res.* **12**(ARTICLE), pp. 2493–2537. Cited page 8

[23] B. C. Csáji *et al.* 2001. "Approximation with artificial neural networks". *Faculty of Sciences, Eötvös Loránd University, Hungary* . Cited page 8

[24] E. de Miguel & G. Jackson 2006. "The nature of the calculation of the pressure in molecular simulations of continuous models from volume perturbations". *J. Chem. Phys.* **125**(16), p. 164 109. Cited page 61

[25] P. Diaconis, G. Lebeau & L. Michel 2011. "Geometric analysis for the Metropolis algorithm on Lipschitz domains". *Invent. Math.* **185**(2), pp. 239–281. Cited page 25

[26] S. S. Du, X. Zhai, B. Poczos & A. Singh 2018. "Gradient descent provably optimizes over-parameterized neural networks". *arXiv preprint arXiv:1810.02054* . Cited page 11

[27] A. Dvoretzky, J. Kiefer & J. Wolfowitz 1956. "Asymptotic Minimax Character of the Sample Distribution Function and of the Classical Multinomial Estimator". *Ann. Math. Stat.* **27**(3), pp. 642–669. Cited pages 49 and 76

[28] B. Efron 1979. "Bootstrap methods: another look at the jackknife". *Ann. Stat.* , pp. 1–26. Cited page 78

[29] M. Engel, J. A. Anderson, S. C. Glotzer, M. Isobe, E. P. Bernard & W. Krauth 2013. "Hard-disk equation of state: First-order liquid-hexatic transition in two dimensions with three simulation methods". *Phys. Rev. E* **87**, p. 042 134. Cited pages 7, 35, 49, 73, 79, and 86

[30] R. Eppenga & D. Frenkel 1984. "Monte Carlo study of the isotropic and nematic phases of infinitely thin hard platelets". *Mol. Phys.* **52**(6), pp. 1303–1334. Cited page 61

[31] J. J. Erpenbeck, W. W. Wood & B. J. Berne 1977. "Statistical Mechanics: Part B: Time-Dependent Processes". *Modern Theoretical Chemistry* **6**. Cited page 62

[32] C. Farabet, C. Couprie, L. Najman & Y. LeCun 2012. "Learning hierarchical features for scene labeling". *IEEE Trans. Pattern Anal. Mach. Intell.* **35**(8), pp. 1915–1929. Cited page 8

[33] L. Fejes 1940. "Über einen geometrischen Satz". *Math. Z.* **46**, pp. 83–85. Cited page 13

[34] Y. Feng & Y. Tu 2021. "The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima". *Proc. Natl. Acad. Sci. U.S.A.* **118**(9). Cited page 11

[35] H. Flyvbjerg & H. G. Petersen 1989. "Error estimates on averages of correlated data". *J. Chem. Phys.* **91**(1), pp. 461–466. Cited pages 75 and 77

[36] R. Ge, F. Huang, C. Jin & Y. Yuan 2015. "Escaping from saddle points—online stochastic gradient for tensor decomposition". In "Conference on learning theory", (PMLR) pp. 797–842. Cited page 11

[37] R. Gower, O. Sebbouh & N. Loizou 2021. "SGD for Structured Nonconvex Functions: Learning Rates, Minibatching and Interpolation". In "International Conference on Artificial Intelligence and Statistics", (PMLR) pp. 1315–1323. Cited page 11

[38] S. Gu, Z. Ghahramani & R. E. Turner 2015. "Neural adaptive sequential Monte Carlo". *Advances in neural information processing systems* **28**. Cited page 8

[39] M. Gurbuzbalaban, U. Simsekli & L. Zhu 2021. "The Heavy-Tail Phenomenon in SGD". In "International Conference on Machine Learning", (PMLR) pp. 3964–3975. Cited page 11

[40] B. I. Halperin & David R. Nelson 1978. "Theory of Two-Dimensional Melting". *Phys. Rev. Lett.* **41**, pp. 121–124. Cited pages 13 and 62

[41] T. Helmuth, W. Perkins & S. Petti 2022. "Correlation decay for hard spheres via Markov chains". *Ann. Appl. Probab.* **32**(3). Cited page 13

[42] S. Hochreiter & J. Schmidhuber 1997. "Flat minima". *Neural Comput.* **9**(1), pp. 1–42. Cited page 11

[43] J. L. Hodges 1958. "The significance probability of the Smirnov two-sample test". *Ark. Mat.* **3**(5), pp. 469 – 486. Cited page 76

[44] L. Hodgkinson & Michael M. W. 2021. "Multiplicative Noise and Heavy Tails in Stochastic Optimization". In "Proceedings of the 38th International Conference on Machine Learning", , edited by Marina Meila & Tong Zhang *Proceedings of Machine Learning Research*, volume 139 (PMLR) pp. 4262–4274. URL https://proceedings.mlr.press/v139/hodgkinson21a.html. Cited page 11

[45] P. Höllmer, A. C. Maggs & W. Krauth 2022. "Hard-disk dipoles and non-reversible Markov chains". *J. Chem. Phys.* **156**(8), p. 084 108. Cited pages 28 and 35

[46] P. Höllmer, N. Noirault, B. Li, A. C. Maggs & W. Krauth 2022. "Sparse Hard-Disk Packings and Local Markov Chains". URL http://dx.doi.org/10.1007/s10955-022-02908-4. Cited pages 3, 6, 7, 13, 17, 22, 23, 24, 25, 27, 28, 29, 33, 35, 36, 41, 42, and 43

[47] K. Hornik, M. Stinchcombe & H. White 1989. "Multilayer feedforward networks are universal approximators". *Neural Netw.* **2**(5), pp. 359–366. Cited pages 2 and 8

[48] E. Ising 1925. "Beitrag zur Theorie des Ferromagnetismus". *Z. Phys.* **31**(1), pp. 253–258. Cited page 16

[49] M. Isobe 1999. "Simple and Efficient Algorithm for Large Scale Molecular Dynamics Simulation in Hard Disk System". *International Journal of Modern Physics C* **10**, pp. 1281–1293. Cited page 4

[50] A. Jaster 1999. "An improved Metropolis algorithm for hard core systems". *Physica A* **264**(1), pp. 134 – 141. Cited pages 7 and 86

[51] A. Jaster 1999. "Computer simulations of the two-dimensional melting transition using hard disks". *Phys. Rev. E* **59**(3), pp. 2594–2602. Cited page 7

[52] A. Jaster 2004. "The hexatic phase of the two-dimensional hard disk system". *Phys. Lett. A* **330**(1), pp. 120 – 125. Cited pages 7 and 86

[53] M. Jonsson 2018. "Standard error estimation by an automated blocking method". *Phys. Rev. E* **98**(4), p. 043 304. Cited pages 75 and 78

[54] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko *et al.* 2021. "Highly accurate protein structure prediction with AlphaFold". *Nature* **596**(7873), pp. 583–589. Cited pages 2 and 8

[55] M. Kahle 2012. "Sparse Locally-Jammed Disk Packings". *Ann. Comb.* **16**(4), pp. 773–780. Cited pages 6, 23, 24, and 25

[56] Tobias A. Kampmann, Horst-Holger Boltz & Jan Kierfeld 2015. "Parallelized event chain algorithm for dense hard sphere and polymer systems". *Journal of Computational Physics* **281**, pp. 864–875. Cited page 5

[57] S. C. Kapfer & W. Krauth 2013. "Sampling from a polytope and hard-disk Monte Carlo". *J. Phys. Conf. Ser.* **454**(1), p. 012 031. Cited pages 5, 36, and 73

[58] H. Karimi, J. Nutini & M. Schmidt 2020. "Linear Convergence of Gradient and Proximal-Gradient Methods Under the Polyak-Łojasiewicz Condition". 1608.04636. Cited page 9

[59] D. P. Kingma & J. Ba 2014. "Adam: A Method for Stochastic Optimization". *International Conference on Learning Representations* . Cited page 10

[60] M. Klement & M. Engel 2019. "Efficient equilibration of hard spheres with Newtonian event chains". *J. Chem. Phys.* **150**(17), p. 174 108. Cited pages 6, 29, 35, and 37

[61] D. E. Knuth 2014. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms* (Addison-Wesley Professional). Cited page 2

[62] W. Krauth 2006. *Statistical Mechanics: Algorithms and Computations* (Oxford University Press). Cited pages 1, 7, 13, 14, 17, 19, 20, 29, and 56

[63] W. Krauth 2021. "Event-Chain Monte Carlo: Foundations, Applications, and Prospects". *Front. Phys.* **9**, p. 229. Cited page 35

[64] A. Krizhevsky, I. Sutskever & G. E. Hinton 2017. "Imagenet classification with deep convolutional neural networks". *Commun. ACM* **60**(6), pp. 84–90. Cited page 8

[65] L. Lamport 1979. "How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs". *IEEE Trans. Comput.* **C-28**(9), pp. 690–691. Cited pages 5 and 66

[66] L. D. Landau & E. M. Lifshitz 2013. *Statistical Physics* volume 5 (Elsevier). Cited page 11

[67] J. L. Lebowitz & O. Penrose 1964. "Convergence of Virial Expansions". *J. Math. Phys.* **5**, pp. 841–847. Cited page 13

[68] Y. LeCun, Y. Bengio & G. Hinton 2015. "Deep learning". *Nature* **521**(7553), pp. 436–444. Cited page 8

[69] D. A. Levin, Y. Peres & E. L. Wilmer 2008. *Markov Chains and Mixing Times* (American Mathematical Society). Cited pages 29 and 32

[70] B. Li, Y. Nishikawa, P. Höllmer, L. Carillo, A. C. Maggs & W. Krauth 2022. "Hard-disk pressure computations—a historic perspective". *J. Chem. Phys.* **157**(23), p. 234 111.
Cited pages 3, 4, 7, 13, 17, 18, 19, 21, 22, 34, 35, 44, 45, 53, 54, 55, 56, 58, 59, 60, 61, 62, 73, 75, 80, 81, 83, 85, and 8

[71] B. Li, S. Todo, A. C. Maggs & W. Krauth 2021. "Multithreaded event-chain Monte Carlo with local times". *Comput. Phys. Commun.* **261**, p. 107 702.
Cited pages 3, 5, 29, 36, 37, 38, 39, 40, 52, 59, 65, 67, 68, 70, and 71

[72] H. Li, Z. Xu, G. Taylor, C. Studer & T. Goldstein 2018. "Visualizing the Loss Landscape of Neural Nets". In "Neural Information Processing Systems", .
Cited page 10

[73] Z. Li, S. Malladi & S. Arora 2021. "On the Validity of Modeling SGD with Stochastic Differential Equations (SDEs)". *arXiv preprint arXiv:2102.12470* .
Cited page 11

[74] K. Liu, L. Ziyin & M. Ueda 2021. "Noise and Fluctuation of Finite Learning Rate Stochastic Gradient Descent". In "Proceedings of the 38th International Conference on Machine Learning", , edited by Marina Meila & Tong Zhang *Proceedings of Machine Learning Research*, volume 139 (PMLR) pp. 7045–7056. URL `https://proceedings.mlr.press/v139/liu21ad.html`. Cited page 11

[75] B. D. Lubachevsky 1991. "How to simulate billiards and similar systems". *J. Comput. Phys.* , pp. 255–283. Cited page 29

[76] B. D. Lubachevsky 1992. "Simulating billiards serially and in parallel". *International Journal in Computer Simulation* **2**, pp. 373–411. Cited page 39

[77] B. D. Lubachevsky & F. H. Stillinger 1990. "Geometric properties of random disk packings". *J. Stat. Phys.* **60**(5), pp. 561–583. Cited page 43

[78] C. H. Mak 2006. "Large-scale simulations of the two-dimensional melting of hard disks". *Phys. Rev. E* **73**, p. 065 104. Cited pages 7 and 86

[79] P. Massart 1990. "The Tight Constant in the Dvoretzky-Kiefer-Wolfowitz Inequality". *Ann. Probab.* **18**(3), pp. 1269–1283. Cited pages 49 and 76

[80] J. E. Mayer & W. W. Wood 1965. "Interfacial Tension Effects in Finite, Periodic, Two-Dimensional Systems". *J. Chem. Phys.* **42**, pp. 4268–4274. Cited page 20

[81] Q. Meng, S. Gong, W. Chen, Z. Ma & T. Liu 2020. "Dynamic of Stochastic Gradient Descent with State-Dependent Noise". *arXiv preprint arXiv:2006.13719* . Cited page 11

[82] N. D. Mermin 1968. "Crystalline Order in Two Dimensions". *Phys. Rev.* **176**, pp. 250–254. Cited page 62

[83] P. Mertikopoulos, N. Hallak, A. Kavis & V. Cevher 2020. "On the Almost Sure Convergence of Stochastic Gradient Descent in Non-Convex Problems". In "Advances in Neural Information Processing Systems", , edited by H. Larochelle,

M. Ranzato, R. Hadsell, M.F. Balcan & H. Lin volume 33 (Curran Associates, Inc.) pp. 1117–1128. URL https://proceedings.neurips.cc/paper/2020/file/0cb5ebb1b34ec343dfe135db691e4a85-Paper. Cited pages 10 and 11

[84] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller & E. Teller 1953. "Equation of State Calculations by Fast Computing Machines". *J. Chem. Phys.* **21**, pp. 1087–1092. Cited pages 3, 7, 29, 33, 61, 84, 85, and 86

[85] N. Metropolis & S. Ulam 1949. "The Monte Carlo Method". *J. Am. Stat. Assoc.* **44**(247), pp. 335–341. Cited page 1

[86] M. Michel 2016. *Irreversible Markov chains by the factorized Metropolis filter: Algorithms and applications in particle systems and spin models*. Ph.D. thesis Ecole Normale Superieure de Paris - ENS Paris. URL https://tel.archives-ouvertes.fr/tel-01394204. Cited page 31

[87] M. Michel, A. Durmus & S. Sénécal 2020. "Forward Event-Chain Monte Carlo: Fast Sampling by Randomness Control in Irreversible Markov Chains". *J. Comput. Graph. Stat.* **29**(4), pp. 689–702. Cited pages 6, 29, 35, and 37

[88] M. Michel, S. C. Kapfer & W. Krauth 2014. "Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps". *J. Chem. Phys.* **140**(5), 054116. Cited pages 53 and 59

[89] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra & M. Riedmiller 2013. "Playing atari with deep reinforcement learning". *arXiv preprint arXiv:1312.5602* . Cited page 8

[90] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.* 2015. "Human-level control through deep reinforcement learning". *Nature* **518**(7540), pp. 529–533. Cited page 8

[91] T. Mori, L. Ziyin, K. Liu & M. Ueda 2022. "Power-law escape rate of SGD". In "Proceedings of the 39th International Conference on Machine Learning", , edited by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu & Sivan Sabato *Proceedings of Machine Learning Research*, volume 162 (PMLR) pp. 15 959–15 975. URL https://proceedings.mlr.press/v162/mori22a.html. Cited page 11

[92] T. Müller, B. McWilliams, F. Rousselle, M. Gross & J. Novák 2019. "Neural importance sampling". *ACM Trans. Graph.* **38**(5), pp. 1–19. Cited page 8

[93] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak & I. Sutskever 2021. "Deep double descent: Where bigger models and more data hurt". *J. Stat. Mech.* **2021**(12), p. 124 003. Cited page 9

[94] Y. Nishikawa, J. Takahashi & T. Takahashi 2021. "Stationary Bootstrap: A Refined Error Estimation for Equilibrium Time Series". 2112.11837 URL http://dx.doi.org/10.48550/ARXIV.2112.11837. Cited pages 75 and 78

[95] L. Onsager 1944. "Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition". *Phys. Rev.* **65**, pp. 117–149. Cited page 1

[96] J. Pach & M. Sharir 2009. *Combinatorial Geometry and Its Algorithmic Applications Mathematical Surveys and Monographs*, volume 152 (American Mathematical Society). Cited page 23

[97] A. Patton, D. N. Politis & H. White 2009. "Correction to "Automatic Block-Length Selection for the Dependent Bootstrap" by D. Politis and H. White". *Econom. Rev.* **28**(4), pp. 372–375. Cited pages 75 and 78

[98] R. Peierls 1936. "On Ising's model of ferromagnetism". *Math. Proc. Camb. Philos. Soc.* **32**, pp. 477–481. Cited page 1

[99] R. Pemantle 1990. "Nonconvergence to unstable points in urn models and stochastic approximations". *Ann. Probab.* **18**(2), pp. 698–712. Cited pages 10 and 11

[100] D. N. Politis & J. P. Romano 1994. "The Stationary Bootstrap". *J. Am. Stat. Assoc.* **89**(428), pp. 1303–1313. Cited pages 75 and 78

[101] D. N. Politis & H. White 2004. "Automatic Block-Length Selection for the Dependent Bootstrap". *Econom. Rev.* **23**(1), pp. 53–70. Cited pages 75 and 78

[102] W. Qi, A. P. Gantapara & M. Dijkstra 2014. "Two-stage melting induced by dislocations and grain boundaries in monolayers of hard spheres". *Soft Matter* **10**(30), p. 5449. Cited pages 7 and 86

[103] Hannes R. 1989. *The Fokker-Planck Equation* (Springer Berlin Heidelberg). URL `http://dx.doi.org/10.1007/978-3-642-61544-3`. Cited page 11

[104] D. C. Rapaport 1980. "The Event Scheduling Problem in Molecular Dynamic Simulation". *J. Comput. Phys.* **34**, pp. 184–201. Cited page 29

[105] S. J. Reddi, S. Kale & S. Kumar 2019. "On the convergence of adam and beyond". *arXiv preprint arXiv:1904.09237* . Cited page 10

[106] T. Richthammer 2007. "Translation-Invariance of Two-Dimensional Gibbsian Point Processes". *Commun. Math. Phys.* **274**(1), pp. 81–122. Cited page 13

[107] T. Richthammer 2016. "Lower Bound on the Mean Square Displacement of Particles in the Hard Disk Model". *Commun. Math. Phys.* **345**, pp. 1077–1099. Cited page 13

[108] H. Robbins & S. Monro 1951. "A stochastic approximation method". *Ann. Math. Statist.* **22**(3), pp. 400–407. Cited pages 10 and 11

[109] S. Ruder 2016. "An overview of gradient descent optimization algorithms". *arXiv preprint arXiv:1609.04747* . Cited pages 10 and 11

[110] J. Schmidhuber 2015. "Deep learning in neural networks: An overview". *Neural Netw.* **61**, pp. 85–117. Cited page 8

[111] O. Sebbouh, R. M. Gower & A. Defazio 2021. "Almost sure convergence rates for stochastic gradient descent and stochastic heavy ball". In "Conference on Learning Theory", (PMLR) pp. 3935–3971. Cited page 10

[112] S. L. Smith & Q. V. Le 2018. "A Bayesian Perspective on Generalization and Stochastic Gradient Descent". In "International Conference on Learning Representations", . Cited page 11

[113] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan & S. Ganguli 2015. "Deep Unsupervised Learning using Nonequilibrium Thermodynamics". In "Proceedings of the 32nd International Conference on Machine Learning", , edited by Francis Bach & David Blei *Proceedings of Machine Learning Research*, volume 37 (PMLR, Lille, France) pp. 2256–2265. URL https://proceedings.mlr.press/v37/sohl-dickstein15.html. Cited page 8

[114] A. Sokal 1997. *Monte Carlo Methods in Statistical Mechanics: Foundations and New Algorithms* (Springer US). ISBN 978-1-4899-0319-8 pp. 131–192. URL http://dx.doi.org/10.1007/978-1-4899-0319-8_6. Cited page 32

[115] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever & R. Salakhutdinov 2014. "Dropout: a simple way to prevent neural networks from overfitting". *J. Mach. Learn. Res.* **15**(1), pp. 1929–1958. Cited page 10

[116] S. Vaswani, F. Bach & M. Schmidt 2019. "Fast and Faster Convergence of SGD for Over-Parameterized Models and an Accelerated Perceptron". In "The 22nd International Conference on Artificial Intelligence and Statistics", (PMLR) pp. 1195–1204. Cited page 9

[117] L. Verlet 1967. "Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules". *Phys. Rev.* **159**(1), pp. 98–103. Cited page 36

[118] S. Vlaski & A. H. Sayed 2019. "Second-Order Guarantees of Stochastic Gradient Descent in Non-Convex Optimization". 1908.07023. Cited pages 10 and 11

[119] L. Wasserman 2004. *All of Statistics* (Springer, New York). URL http://dx.doi.org/10.1007/978-0-387-21736-9. Cited pages 48, 49, 75, 76, and 77

[120] R. F. B. Weigel 2018. "Equilibration of orientational order in hard disks via arcuate event-chain Monte Carlo". Master thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg URL https://theorie1.physik.uni-erlangen.de/research/theses/2018-ma-roweigel.html. Cited page 36

[121] D. B. Wilson 2000. "How to couple from the past using a read-once source of randomness". *Random Structures & Algorithms* **16**(1), pp. 85–113. Cited page 29

[122] K. G. Wilson 1975. "The renormalization group: Critical phenomena and the Kondo problem". *Rev. Mod. Phys.* **47**, pp. 773–840. Cited page 77

[123] S. Wojtowytsch 2021. "Stochastic gradient descent with noise of machine learning type. Part I: Discrete time analysis". *arXiv preprint arXiv:2105.01650* . Cited pages 9 and 11

[124] S. Wojtowytsch 2021. "Stochastic gradient descent with noise of machine learning type. Part II: Continuous time analysis". *arXiv preprint arXiv:2106.02588* . Cited page 11

[125] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.* 2020. "Transformers: State-of-the-art natural language processing". In "Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations", pp. 38–45. Cited page 8

[126] S. Wolfram 2003. *The mathematica book* volume 1 (Wolfram Research, Inc.). Cited page 1

[127] W. W. Wood 1968. "Monte Carlo Calculations for Hard Disks in the Isothermal-Isobaric Ensemble". *J. Chem. Phys.* **48**, pp. 415–434. Cited page 17

[128] W. W. Wood 1970. "NpT-ensemble Monte Carlo calculations for the hard-disk fluid". *J. Chem. Phys.* **52**(2), pp. 729–741. Cited page 7

[129] W. W. Wood 1970. "NpT-ensemble monte carlo calculations for the hard-disk fluid". *J. Chem. Phys.* **52**, pp. 729–741. Cited page 17

[130] W. W. Wood, J. J. Erpenbeck, G. A. Baker & J. D. Johnson 2000. "Molecular dynamics ensemble, equation of state, and ergodicity". *Phys. Rev. E* **63**, p. 011 106. Cited pages 18 and 62

[131] Z. Xie, I. Sato & M. Sugiyama 2021. "A Diffusion Theory For Deep Learning Dynamics: Stochastic Gradient Descent Exponentially Favors Flat Minima". In "International Conference on Learning Representations", URL https://openreview.net/forum?id=wXgk_iCiYGo. Cited page 11

[132] C. Xing, D. Arpit, C. Tsirigotis & Y. Bengio 2018. "A Walk with SGD". Cite arxiv:1802.08770Comment: First two authors contributed equally URL http://arxiv.org/abs/1802.08770. Cited page 11

[133] C. Zhang, Q. Liao, A. Rakhlin, B. Miranda, N. Golowich & T. Poggio 2018. "Theory of deep learning IIb: Optimization properties of SGD". *arXiv preprint arXiv:1801.02254* . Cited page 11

[134] L. Ziyin, K. Liu, T. Mori & M. Ueda 2021. "Strength of Minibatch Noise in SGD". *arXiv preprint arXiv:2102.05375* . Cited page 11

[135] J. A. Zollweg & G. V. Chester 1992. "Melting in two dimensions". *Phys. Rev. B* **46**, pp. 11 186–11 189. Cited pages 7 and 86

[136] D. Zou, J. Wu, V. Braverman, Q. Gu & S. Kakade 2021. "Benign overfitting of constant-stepsize sgd for linear regression". In "Proceedings of Thirty Fourth

Conference on Learning Theory", , edited by Mikhail Belkin & Samory Kpotufe *Proceedings of Machine Learning Research*, volume 134 (PMLR) pp. 4633–4635. URL https://proceedings.mlr.press/v134/zou21a.html. Cited page 11

Publications

Publication 1: Multithreaded event-chain Monte Carlo with local times

Computer Physics Communications 261 (2021) 107702

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Multithreaded event-chain Monte Carlo with local times☆,☆☆

Botao Li [a], Synge Todo [b,c], A.C. Maggs [d], Werner Krauth [a,*]

[a] Laboratoire de Physique de l'Ecole normale supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université Paris-Diderot, Sorbonne Paris Cité, Paris, France
[b] Department of Physics, University of Tokyo, 113-0033 Tokyo, Japan
[c] Institute for Solid State Physics, University of Tokyo, 277-8581Kashiwa Japan
[d] CNRS UMR7083, ESPCI Paris, PSL Research University, 10 rue Vauquelin, 75005 Paris, France

ARTICLE INFO

Article history:
Received 27 May 2020
Received in revised form 8 October 2020
Accepted 14 October 2020
Available online 1 November 2020

Keywords:
Monte Carlo algorithm
Irreversible Markov chain
Multithreading
Event-chain algorithm
Sequential consistency model
C++
Python
Fortran90

ABSTRACT

We present a multithreaded event-chain Monte Carlo algorithm (ECMC) for hard spheres. Threads synchronize at infrequent breakpoints and otherwise scan for local horizon violations. Using a mapping onto absorbing Markov chains, we rigorously prove the correctness of a sequential-consistency implementation for small test suites. On x86 and ARM processors, a C++ (OpenMP) implementation that uses compare-and-swap primitives for data access achieves considerable speed-up with respect to single-threaded code. The generalized birthday problem suggests that for the number of threads scaling as the square root of the number of spheres, the horizon-violation probability remains small for a fixed simulation time. We provide C++ and Python open-source code that reproduces all our results.

Program summary
Program title: ParaSpheres.
CPC Library link to program files: https://doi.org/10.17632/c3rjk5k3z9.1
Developer's repository link: https://github.com/jellyfysh/ParaSpheres
Licensing provisions: GNU GPLv3.
Programming languages: Python 3, C++, Fortran90.
Nature of problem: Multithreaded event-chain Monte Carlo for hard spheres.
Solution method: Event-driven irreversible Markov-chain Monte Carlo algorithm using local times.
Additional comments: The collection of programs is complete with shell scripts that allow one to reproduce all data, and all the figures of the paper. Change of density and system size is straightforward. The manuscript is accompanied by a frozen copy of the GitHub repository that is made publicly available on GitHub (repository https://github.com/jellyfysh/ParaSpheres, commit hash e2aa5b9727fb080ebe65581586c0f6133efa495d).

1. Introduction

Event-chain Monte Carlo (ECMC) [1,2] is an event-driven realization of a continuous-time irreversible Markov chain that has found applications in statistical physics [3,4] and related fields [5]. Initially restricted to hard spheres and to models with piece-wise constant pair potentials [6], ECMC was subsequently extended to continuous potentials, such as spin models and all-atom particle systems with long-range interactions [7,8]. Potentials need not be pair-wise additive [9]. In opposition to standard Monte Carlo methods, such as the Metropolis algorithm [10], ECMC does not evaluate the potential $U(\mathbf{x})$ of a configuration $\mathbf{x}$ (nor any ratio of potentials) in order to sample the Boltzmann distribution $\pi(\mathbf{x}) = \exp[-\beta U(\mathbf{x})]$, with inverse temperature β.

For hard spheres, ECMC is a special case of event-driven molecular dynamics [11,12]. In molecular dynamics, usually all N spheres have non-zero velocities, and the number of candidate collision events at any time is $\mathcal{O}(N)$. A central scheduler, efficiently implemented through a heap data structure, yields the next collision with computational effort $\mathcal{O}(1)$, and it updates the heap in at most $\mathcal{O}(\log N)$ operations [13,14]. Event times are global, and the CPU clock advances together with the collision times. The global collision times and the required communications at events complicate multithread implementations [15–19]. Domain decomposition, another strategy to cope with synchronization, is also problematic [20].

☆ The review of this paper was arranged by Prof. D.P. Landau.
☆☆ This paper and its associated computer program are available via the Computer Physics Communication homepage on ScienceDirect (http://www.sciencedirect.com/science/journal/00104655).
* Corresponding author.
E-mail address: werner.krauth@ens.fr (W. Krauth).

https://doi.org/10.1016/j.cpc.2020.107702
0010-4655/

In hard-sphere ECMC, a set $\mathcal{A}_t$ of $k < N$ "active" spheres (all of radius σ) have the same non-zero velocity $\mathbf{v}$ that changes infrequently. All other spheres are "static". At a lifting [21] $l_t = ([i \to j], (\mathbf{x}, \mathbf{x}'), t)$, an active sphere i collides at time t with a target sphere j at contact ($|\mathbf{x}' - \mathbf{x}| = 2\sigma$, a condition that must be adapted for periodic boundary conditions). The lifting l_t connects an in-state (the configuration just before time t, at time t^-) with an out-state (the configuration just after time t, at time t^+):

$$\text{in-state}: \begin{bmatrix} i \in \mathcal{A}_{t^-}, \quad j \notin \mathcal{A}_{t^-} \\ \mathbf{x}_i(t^-) = \mathbf{x} \\ \mathbf{x}_j(t^-) = \mathbf{x}' \\ \mathbf{v}_i(t^-) = \mathbf{v} \\ \mathbf{v}_j(t^-) = 0 \end{bmatrix}; \quad \text{out-state}: \begin{vmatrix} i \notin \mathcal{A}_{t^+}, \quad j \in \mathcal{A}_{t^+} \\ \mathbf{x}_i(t^+) = \mathbf{x} \\ \mathbf{x}_j(t^+) = \mathbf{x}' \\ \mathbf{v}_i(t^+) = 0 \\ \mathbf{v}_j(t^+) = \mathbf{v} \end{vmatrix}. \quad (1)$$

We consider in this paper two-dimensional spheres in a square box with periodic boundary conditions. In this system, the direction of $\mathbf{v}$ must be changed at certain breakpoints for the algorithm to be irreducible [22]. However, we restrict our attention to ECMC in between two such breakpoints h and h' with, for concreteness, $\mathbf{v} = (v_x, v_y) = (1, 0)$. For a generic "lifted" [21] initial configuration $\{\mathcal{C}_h, \mathcal{A}_h\}$ at h, ECMC is deterministic up to h'. Generically no two liftings take place at the same time t, so that they can be identified by their time.

Our multithreaded ECMC algorithm propagates $k = |\mathcal{A}|$ active spheres in independent threads, with shared memory. In between h and h', it only uses local-time attributes of each sphere. At a lifting $l_{t_i} = ([i \to j], (\mathbf{x}, \mathbf{x}'), t_i)$, j synchronizes with i (the local time t_j is set equal to t_i). For a sphere i to move, it must not violate certain horizon conditions of nearby spheres j. In the absence of horizon violations between h and h', multithreaded ECMC will be proven equivalent to the global-time process.

The motivation for our work is twofold. First, we strive to speed up current hard-sphere simulations where, typically, $N \sim 1 \times 10^6$. These simulations require weeks or months of run time to decorrelate from the initial configuration [3,23]. Using a connection to the generalized birthday problem in mathematics, we will argue that such simulations can successfully run with $k \lesssim \sqrt{N}$. Our approach to multithreading thus uses the freedom to tune the number of active spheres. Second, by providing proof of concept for multithreaded ECMC algorithms, we hope to motivate the development of parallel ECMC algorithms for other system to which sequential ECMC applies already.

The multithreaded ECMC algorithm is presented in two versions. One implementation uses the sequential-consistency model [24]. Mapped onto an absorbing Markov chain, its correctness is rigorously proven for small test suites. The C++ implementation uses OpenMP to map active chains onto hardware threads, together with atomic primitives [25] for fine-grained control of interactions between threads. Considerable speed-up with respect to a single-threaded version is achieved. The few simultaneously moving spheres ($k \ll N$) avoid communication bottlenecks between threads, even though each hard-sphere lifting involves only little computation.

Subtle aspects of our algorithm surface through the confrontation of the C++ implementation with the sequential-consistency computational model on the same test suites. By reordering single statements in the code, we may for example introduce rare bugs that are not detected during random testing, but that are readily exhibited in the rigorous solution, and that illustrate difficulties stemming from possible compiler or processor re-ordering.

Code availability. *Cell-based ECMC for two-dimensional hard spheres is implemented (in Fortran90) as* `CellECMC.f90`*. Our version is slightly modified from two original programs made available in a* `Fortran90/Historic` *directory, written by E. P. Bernard (see Acknowledgments). The code prepares initial configurations. It is used in validation scripts.*

2. Algorithms: from global-time processes to multithreaded ECMC

In this section, we start with the definition of a continuous-time process, Algorithm 1, that is manifestly equivalent to molecular dynamics with the collision rules of Eq. (1). Its event-driven version, Algorithm 2, provides the reference set $\mathcal{L}^{\text{ref}}$ of liftings used in our validation scripts (see Section 3). The single-threaded Algorithm 3 relies on local times. It has correct output if no horizon violation takes place. Its event-driven version, Algorithm 4, yields a practical method that can be implemented and tested. Algorithms 5 and 6 realize multithreaded ECMC, the latter in a highly efficient C++ implementation.

2.1. Continuous processes and ECMC with global time

Algorithm 1 (*Continuous Process with Global Time*)**.** At global time $t = h$, an initial lifted configuration $\{\mathcal{C}_h, \mathcal{A}_h\}$ is given ($\mathbf{v}_i = \mathbf{v} = (1, 0)\ \forall i \in \mathcal{A}_h$ and $\mathbf{v}_i = 0\ \forall i \notin \mathcal{A}_h$). All spheres i carry local times t_i, with, initially, $t_i(h) = h\ \forall i$. For active spheres ($i \in \mathcal{A}_t$), $\mathrm{d}t_i/\mathrm{d}t = 1$. At a lifting $l_t = ([i \to j], (\mathbf{x}, \mathbf{x}'), t)$ the local time of sphere j is updated as $t_j(t^+) = t$ and, furthermore, $\mathcal{A}_{t^+} = \mathcal{A}_{t^-} \setminus \{i\} \cup \{j\}$. The algorithm stops at global time $t = h'$, and outputs the lifted configuration $\{\mathcal{C}_{h'}, \mathcal{A}_{h'}\}$, and the set $\mathcal{L}_{h'} = \{l_t : h < t < h'\}$ of liftings that have taken place between h and h'.

Remark 1 (*Meaning of Local Times*)**.** In Algorithm 1, the local time $t_i(t)$ is a function of the global time t. It gives the global time at which sphere i was last active (or $t_i(t) = h$ if i was not active for $[h, t]$). Therefore $t_i(t) = t\ \forall i \in \mathcal{A}_t$ and $t_i(t) < t\ \forall i \notin \mathcal{A}_t$.

Remark 2 (*Positivity of Local-time Updates*)**.** In Algorithm 1, at any lifting l_t, the update of t_j is positive: $t_j(t^+) - t_j(t^-) > 0$.

Remark 3 (*Time-reversal Invariance*)**.** Algorithm 1 is deterministic and time-reversal invariant: If an initial lifted configuration $\{\mathcal{C}_h, \mathcal{A}_h\}$ generates the final lifted configuration $\{\mathcal{C}_{h'}, \mathcal{A}_{h'}\}$ with $\mathbf{v}$, then the latter will reproduce the initial configuration with $-\mathbf{v}$. The set $\mathcal{L}$ of liftings is the same in both cases (with exchanged i and j).

In order to converge towards a given probability distribution, Markov-chain algorithms must satisfy the global-balance condition. It states that the probability flow into a configuration $\mathcal{C}$ (summed over all liftings $\mathcal{A}$) must equal the probability flow out of it [22]. ECMC balances these flows for each lifting individually (for the uniform probability distribution).

Lemma 1. *Algorithm 1 satisfies the global-balance condition for any lifted configuration $\{\mathcal{C}, \mathcal{A}\}$. All lifted configurations accessible from a given initial configuration thus have the same statistical weight.*

Proof. The algorithm is equivalent to molecular dynamics that conserves one-dimensional momenta as well as the energy. The claimed property follows for Algorithm 1 because it is satisfied by molecular dynamics. The property can be shown directly for a discretized version of Algorithm 1 on a rectangular grid aligned with $\mathbf{v}$ with infinitesimal cell size such that each lifted configuration $\{\mathcal{C}, \mathcal{A}\}$ has a unique predecessor. The flow into each lifted configuration equals one. This is equivalent to global balance for the uniform probability distribution. □

The event-driven version of Algorithm 1 is the following:

Algorithm 2 (*ECMC with Global Time*)**.** With input as in Algorithm 1, in each iteration $I = 1, 2, \ldots$, the next global lifting time is computed as $t_{I+1} = t_I + \min_{i\in\mathcal{A}, j\notin\mathcal{A}} \tau_{ij}$,[1] where τ_{ij} is the time of flight from sphere i to sphere j. At time t_I, local times as well as positions of active spheres are advanced to $\tilde{t} = \min\left(t_{I+1}, h'\right)$, and to $\mathbf{x}_i(t_{I+1}) = \mathbf{x}_i(t_I) + (\tilde{t} - t_I)\mathbf{v}$, respectively. If $\tilde{t} = t_{I+1}$ (a lifting $l_{I+1} = (\{i, j\}, \{\mathbf{x}, \mathbf{x}'\}, t_{I+1})$ takes place), the set of active spheres is updated as $\mathcal{A}_{I+1} = \mathcal{A}_I \setminus \{i\} \cup \{j\}$. Otherwise $\tilde{t} = h'$, and the algorithm stops. Output is as in Algorithm 1.

Code availability. *Algorithm 2 is implemented in* `GlobalTimeECMC.py` *and invoked in several validation scripts, for which it generates the reference lifting sets* $\mathcal{L}^{\text{ref}}$.

2.2. Single-threaded processes and ECMC with local times

Algorithm 3, that we now describe, is a single-threaded emulation of our multithreaded Algorithms 5 and 6. A randomly sampled active chain $\iota \in \{1, 2, \ldots\}$ advances (in what corresponds to a thread) for an imposed duration, at most until its local time reaches h'. On thread ι, the active sphere i must remain above the horizons of its neighboring spheres j (see Fig. 1a). The horizon condition is

$$t_i + \tau_{ij} > t_j, \tag{2}$$

where the time of flight is $\tau_{ij} = x_j - x_i + b_{ij}$, with b_{ij} the contact separation parallel to $\mathbf{v}$ between spheres i and j. The horizon condition must be checked for at most three spheres j for a given i because all other spheres are either too far for lifting with i in the direction perpendicular to $\mathbf{v}$ or are prevented from lifting with i by other spheres (see Section 3.1). The algorithm aborts if a horizon violation is encountered. The active chain ι stops if a lifting would be to a sphere j that is itself active. The active chain $\iota + 1$ is then started.

Remark 4 (*Double Role of Horizon Condition*)**.** The horizon condition of Eq. (2) has two roles. First, it is a necessary condition for a lifting of i with j (if it effectively takes place) to produce the required positive local-time update of t_j at the lifting time t (see Remark 2 and Fig. 1a). Second, it is a sufficient non-crossing condition for any sphere k, ensuring that k was not at a previous local time in conflict with i (see Fig. 1b).

It is for the second role discussed in Remark 4 that the horizon condition is checked for all neighboring spheres j of an active sphere i.

Remark 5 (*False Alarms from Horizon Condition*)**.** The horizon condition may lead to false alarms (see Fig. 1b), which could be avoided through the use of the non-crossing condition. The latter is more difficult to check, as it requires the history of past liftings. Our algorithms only implement the horizon condition.

Algorithm 3 (*Single-threaded Continuous Process with Local Times*)**.** With input as in Algorithm 1, active chain $\iota = 1, 2, \ldots$ is initialized (sequentially) with an active sphere i, sampled from $\tilde{\mathcal{A}} = \{i \in \mathcal{A}, t_i \neq h'\}$, and for a local-time interval $\tau_\iota^{\max} = \min\left(\text{ran}, h' - t_i\right)$, where ran is a positive random number. In active chain ι, the active sphere i moves with velocity $\mathbf{v}$ for $\tau \in [0, \tau_\iota^{\max}]$ and

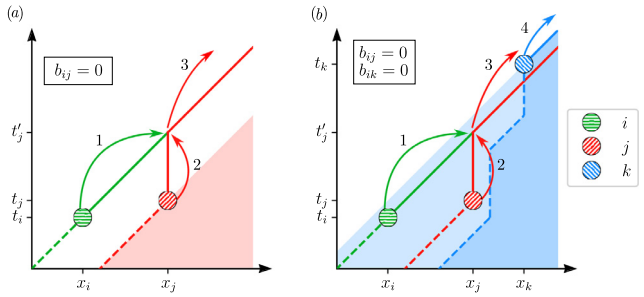

Fig. 1. Horizon condition and non-crossing condition in local-time algorithms. (a) Sphere i is above the horizon of sphere j (shaded area), so that at the lifting of i with j, the local-time update of t_j is positive. (b) Sphere i does not lift with k. It does not cross the past trajectory of k, although it violates the horizon condition with k (light shading) (supposing $b_{ik} = 0$). The lifting of i with j could in principle be allowed under the non-crossing condition with k (dark blue shading), supposing $\tau_{jk} = \infty$.

$\mathrm{d}t_i/\mathrm{d}\tau = 1$, if the horizon condition of Eq. (2) is satisfied for all spheres j. (In case of a horizon violation, the algorithm aborts.) If a lifting $l_{t_i} = ([i{\to}j], (\mathbf{x}, \mathbf{x}'), t_i)$ concerns an active sphere j, the active chain ι stops with i at $\mathbf{x}$. Otherwise, the local time of sphere j is updated as $t_j(t_i^+) = t_i$ and $\mathcal{A} = \mathcal{A} \setminus \{i\} \cup \{j\}$, with the active chain ι now moving j. The algorithm terminates if $\tilde{\mathcal{A}} = \emptyset$, that is, if all the active spheres are stalled. Output is as for Algorithm 1.

Remark 6 (*Stalled Spheres*)**.** "Stalled" spheres (active spheres i with $t_i = h'$) make up the set $\mathcal{A} \setminus \tilde{\mathcal{A}}$. Considering stalled spheres separately simplifies the sampling of $\tilde{\mathcal{A}}$ and the restart from h' for the next leg of the ECMC run.

Lemma 2. *If Algorithm 3 terminates without a horizon violation, its output is identical to that of Algorithm 1.*

Proof. We consider the final lifted configuration $\{\mathcal{C}_{h'}, \mathcal{A}_{h'}\}$ of a run that has terminated without a horizon violation, and that has preserved a log of all local-time updates. The termination condition is $\tilde{\mathcal{A}} = \emptyset$, so that all active spheres are stalled with local time h'. We further consider the final lifting $l_{h''}$ in $\mathcal{L}_{h'}$ (so that $t < h'' \ \forall\ l_t \in \mathcal{L}_{h''}$). Local times of static spheres satisfy $t_i \leq h'' \ \forall i \notin \mathcal{A}_{h'}$. When backtracking, using Algorithm 1 with $-\mathbf{v}$, from h' to h''^+, no lifting takes place among active spheres (see Fig. 2a). The area swept out by the active spheres cannot overlap with a static sphere j because it must have $t_j < h''$ (local times are smaller than the last lifting) and, on the other hand, $t_j > h''$, because of Eq. (2) (see Fig. 2b). The lifting $l_{h''} = ([i{\to}j], (\mathbf{x}, \mathbf{x}'), h'')$ can now be undone. (From $j \in \mathcal{A}_{h'}$ and $i \notin \mathcal{A}_{h'}$, we obtain $\mathcal{A}_{h''} = \mathcal{A}_{h'} \setminus \{j\} \cup \{i\}$. The updated local time $t_j(h''^-)$ can be reconstructed from the log. It is smaller than h''. The lifting is then itself eliminated: $\mathcal{L}_{h''} = \mathcal{L}_{h'} \setminus \{l_{h''}\}$.) All active spheres at h''^- now have local time h''. Similarly, all liftings can be undone, effectively running Algorithm 2 with $-\mathbf{v}$ from h' to h. As Algorithm 1 is time-inversion invariant, the local times at its liftings are the same as those of Algorithm 3. □

For concreteness, in the following event-driven formulation of Algorithm 3, the local-time interval $\tau_\iota^{\max}$ of an active chain ι is chosen equal to the time of flight towards the next lifting.

Algorithm 4 (*Single-threaded ECMC with Local Times*)**.** With input as in Algorithm 1, for each (sequential) active chain $\iota = 1, 2, \ldots$, an active sphere i is sampled from $\tilde{\mathcal{A}} = \{i \in \mathcal{A}, t_i \neq h'\}$. The horizon conditions of Eq. (2) are checked for all[2] spheres j that

[1] τ_{ij} is infinite if i cannot lift with j for the given initial configuration $\mathcal{C}$ and velocity $\mathbf{v}$. The presence of an arrow $[i{\to}j]$ in the directed constraint graph $\mathcal{G}$ indicates that τ_{ij} can be finite (see Section 3.1).

[2] At most three spheres j can have finite τ_{ij} for any i, see Section 3.1.

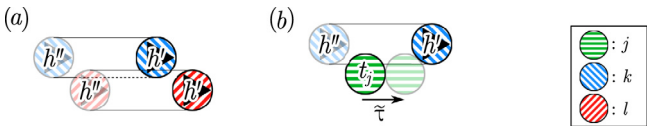

Fig. 2. Backtrack using Algorithm 1 from the final configuration of Algorithm 3. (a) Active spheres k and l do not lift among each other. (b) A static sphere j crossing the trajectory of active sphere k. This crossing is impossible because of the horizon condition ($\tilde{\tau} < h' - h''$ leading to $t_j > h''$, in contradiction with the condition $t_j < h''$).

can have finite time of flight τ_{ij}. The algorithm aborts if a violation occurs. Otherwise, i is moved forward to min $\left(t_i + \tau_{ij}, h'\right)$, and the local time of i and j are updated to that time. The active chain stops if j is an active sphere or if the local time equals h'. Otherwise, the move corresponds to a lifting $l_{t_i+\tau_{ij}} = ([i \to j], (\mathbf{x}, \mathbf{x}'), t_i + \tau_{ij})$ and $\mathcal{A} = \mathcal{A} \setminus \{i\} \cup \{j\}$, with the active chain now moving j. The algorithm terminates if $\tilde{\mathcal{A}} = \emptyset$. Output is as for Algorithm 1.

Code availability. *Algorithm 4 is implemented in* `SingleThreadLocalTimeECMC.py` *and tested in the* `PValidateECMC.sh` *script.*

Remark 7 (*Partial Validation*)**.** In Section 3.2, a variant of Algorithm 4 is used to validate part of a run, even if it does not terminate correctly. When a sphere i detects a horizon violation, its time t_i is recorded. At h', the set $\mathcal{L}_{t^*}$ of all liftings up to the earliest horizon violation, at t^*, agrees with the corresponding partial list of liftings for Algorithm 1.

2.3. Multithreaded ECMC (sequential-consistency model)

Algorithm 5, the subject of the present section, is a model shared-memory ECMC on k threads, that is, on as many threads as there are active spheres. The algorithm adopts the sequential-consistency model [24]. We rigorously prove its correctness for small test suites by mapping the multithreading stage of this algorithm to an absorbing Markov chain. The algorithm allows us to show that certain seemingly innocuous modifications of Algorithm 6 (the C++ implementation) contain bugs that are too rare to be detected by routine testing.

The algorithm has three stages. In the (sequential) initialization stage, it inputs a lifted initial configuration and maps each active sphere to a thread. This is followed by the multithreading stage, where each active chain progresses independently, checking the horizon conditions in its local environment. The algorithm concludes with the (sequential) output stage.

At each step of the multithreading stage of Algorithm 5, a switch randomly selects one of the k statements (one for each thread $a, b, \ldots$) contained in a buffer as $\{\text{next}_a, \text{next}_b, \ldots\}$. The selected statement is executed on the corresponding thread, and then the buffer is updated. The random sequence of statements mimics the absence of thread synchronization except at breakpoints. All threads possess an absorbing **wait** statement. When it is reached throughout, the algorithm progresses to the output stage, followed by successful termination. The program aborts when a thread detects a horizon violation. For our test suites, we prove by explicit construction that each state is connected to at least one of the absorbing states, but we lack a general proof of validity for arbitrary configurations and general N.

In Algorithm 5, each sphere has three attributes, namely a tag, a local time and a position. The sphere's tag indicates whether it is active on a thread ι, stalled, or static. All threads have read/write access to the attributes of all spheres. A state of the Markov chain is constituted by the spheres with their attributes, some local variables and the buffer content.

Algorithm 5 (*Multithreaded ECMC (Sequential-consistency Model)*)**.** At breakpoint $h = 0$, a lifted initial configuration $\{\mathcal{C}_h, \mathcal{A}_h\}$ is input (see Fig. 3 for the example with four spheres). All local times are set to $h = 0$, all tags are put to static, except for the active spheres, whose tags correspond to their thread ι. The buffer is set to $\{1_1, \ldots, 1_\iota, \ldots, 1_k\}$. A random switch selects one buffer element. The corresponding statement is executed on its thread, and the buffer is replenished. The following provides pseudo-code for the multithreading stage (i_ι is the active sphere, j_ι the target sphere, and $distance_\iota$ the difference between h' and the local time, all on thread ι):

```
1ι   τ_ι ← distance_ι; j_ι ← i_ι; x_ι ← ∞
2ι   for j̃ in {1, 2, ..., n} \ i_ι :
3ι     x_j̃ ← j̃.x
4ι     τ_ij̃ ← x_j̃ − i_ι.x − b_ij̃
5ι     if i_ι.t + τ_ij̃ < j̃.t : abort
6ι     if τ_ij̃ < τ_ι :
7ι       j_ι ← j̃
8ι       x_ι ← x_j̃
9ι       τ_ι ← τ_ij̃
10ι  j_ι.tag.CAS(static, ι)
11ι  if j_ι.tag = ι :
12ι    if τ_ι < distance_ι :
13ι      if x_ι = j_ι.x :
14ι        j_ι.t ← i_ι.t + τ_ι
15ι        i_ι.t ← i_ι.t + τ_ι
16ι        i_ι.x ← i_ι.x + τ_ι
17ι        i_ι.tag ← static
18ι        distance_ι ← distance_ι − τ_ι
19ι        i_ι ← j_ι
         else :
20ι        j_ι.tag ← static
           goto 1
       else :
21ι      i_ι.t ← i_ι.t + τ_ι
22ι      i_ι.x ← i_ι.x + τ_ι
23ι      distance_ι ← 0
24ι      i_ι.tag ← stalled
     else : goto 1
25ι  if distance_ι > 0 : goto 1
26ι  wait
```

When all k threads have reached their **wait** statements, the algorithm proceeds to its output stage. Output is as for Algorithm 1.

Code availability. `SequentialMultiThreadECMC.py` *implements Algorithm 5. It also constructs all states connected to the initial state and traces them to the absorbing states. It is called by* `SequentialC4.sh` *and* `SequentialC5.sh`

Remark 8 (*Illustration of Pseudocode*)**.** The multithreading stage of Algorithm 5 corresponds to k identical programs running independently. In the sequential-consistency model, the space of programming statements is thus k-dimensional (one sequence $(1\iota, \ldots, 26\iota)$ per thread), and each displacement in this space proceeds along a randomly chosen coordinate axis. As an example, if for a buffer $\{next_1, \ldots, next_\iota = 20\iota, \ldots, next_k\}$ the switch selects thread ι, then the tag of target particle j_ι is set to "static", and the buffer is updated to $\{next_1, \ldots, next_\iota = 1\iota, \ldots, next_k\}$. The thread ι will thus be restarted at its next selection.

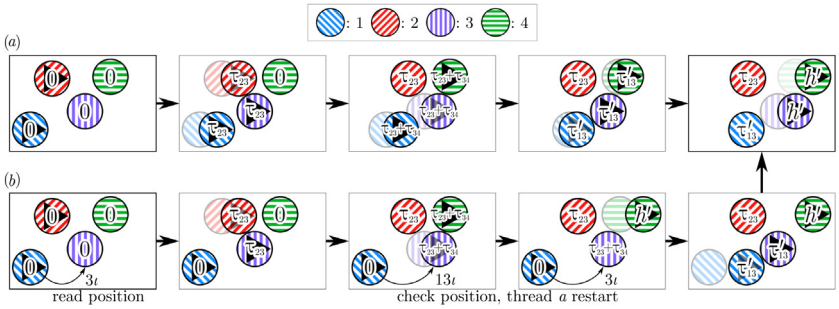

Fig. 3. Algorithm 5, as applied to the SequentialC4 test suite. (a) Reference set $\mathcal{L}^{\text{ref}}$ from Algorithm 2. (b) Run of Algorithm 5 involving a "lock-less" lock rejection on the position of sphere 3.

The compare-and-swap (CAS) statement in 10ι of Algorithm 5 amounts to a single-line **if**. It is equivalent to: "**if** $j_\iota.tag = static$ **:** $j_\iota.tag = \iota$" (if j is static, then it is set to active on thread ι (see Remark 9 for a discussion).

We prove correctness of Algorithm 5, for the SequentialC4 test suite with $N = 2$ and $k = 2$ (see Fig. 3), that we later extend to the SequentialC5 test suite with $N = 5$.

Lemma 3. *If Algorithm 5 terminates without a horizon violation, its output (for the* SequentialC4 *test suite) is identical to that of Algorithm 1.*

Proof. In the SequentialC4 test suite with threads "a" and "b", we suppose that the switch samples a and b with equal probabilities. The two-thread stage of Algorithm 5 then consists in a finite Markov chain with 3670 states S_n that are accessible from the initial state. The **abort** state has no buffer content. All other 3669 states comprise the buffer $\{\text{next}_a, \text{next}_b\}$, the sphere objects (the spheres and their attributes: tag, local time, position), and some thread-specific local variables. One iteration of the Markov chain (selection of next_a or next_b, execution of the corresponding statement, buffer update) realizes the transition from S_n to a state S_m with probability $T_{nm} = 1/2$. The 3670×3670 transition matrix $T = (T_{nm})$ has unit diagonal elements for the **abort**, and for the unique **terminate** state with buffer $\{26a, 26b\}$, which are both absorbing states of the Markov chain. Furthermore, we can show explicitly that all 3670 states have a finite probability to reach an absorbing state in a finite number of steps. This proves that the Markov chain is absorbing. For an absorbing Markov chain, all states that are not absorbing are transient, and they die out at large times. The algorithm thus either ends up in the unique **terminate** state that corresponds to successful completion, or else in the **abort** state. □

All states of the Markov chain may be projected onto their buffer $\{\text{next}_a, \text{next}_b\}$ and visualized (see Fig. 4)).

Remark 9 (*CAS Statement*)**.** The CAS statements (see 10ι in Algorithm 5) acquire their full meaning in the multithreaded Algorithm 6 The way in which they differ from simple **if** statements can already be illustrated in the simplified setting. We suppose two threads a and b. Then, with j_ι the target sphere on thread ι, a buffer content $\{10a, 10b\}$:

⋮	⋮
$10a$	$j_a.tag.\text{CAS}(static, a)$
$11a$	**if** $j_a.tag = a$ **:**
⋮	⋮

⋮	⋮
$10b$	$j_b.tag.\text{CAS}(static, b)$
$11b$	**if** $j_b.tag = b$ **:**
⋮	⋮

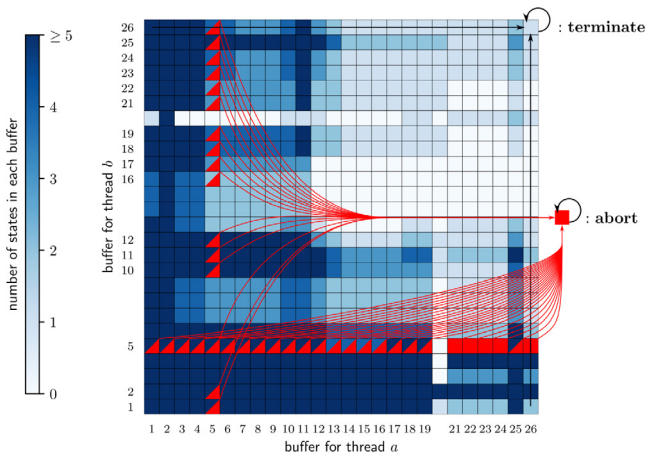

Fig. 4. The 3670 states in Algorithm 5 for the SequentialC4 test suite projected onto the buffer content $\{\text{next}_a, \text{next}_b\}$ (see Fig. 3). The **terminate** buffer $\{26a, 26b\}$ corresponds to a single state.

can belong to a state with $j_a = 3 = j_b = 3$.[3] If the statement $10a$ is selected, sphere 3 becomes active on thread a (through the statement $3.tag = a$). In contrast, if the switch selects $10b$, sphere 3 becomes active on thread b. The program continues consistently for both switch choices, because the selection is made in a single ("atomic") step on each thread and because the sequential-consistency model avoids conflicting memory assignments. In contrast, if the switch selection from $\{10a, 10b\}$ is split as:

⋮	⋮
$10'a$	**if** $j_a.tag = static$ **:**
$10''a$	$j_a.tag = a$
$11a$	**if** $j_a.tag = a$ **:**
⋮	⋮

⋮	⋮
$10'b$	**if** $j_b.tag = static$ **:**
$10''b$	$j_b.tag = b$
$11b$	**if** $j_b.tag = b$ **:**
⋮	⋮

the sequence $10'a \to 10'b \to 10''a \to 11a \to 10''b \to 11b$ results in sphere 3 first becoming active on thread a (and the thread continuing as if this remained the case), and then on thread b, which is inconsistent. In Algorithm 6, the C++ implementation of multithreaded ECMC, the CAS likewise keeps this selection step atomic, and likewise excludes memory conflicts among all threads during this step. It thus plays the role of a lightweight memory lock.

Algorithm 5 features lock-free programming, which is also a key ingredient of Algorithm 6.

[3] This corresponds to the lifted configuration of Fig. 3h.

Remark 10 (*Lock-free Programming*)**.** To illustrate lock-free programming in Algorithm 5, we consider two threads, a and b.

$7a$ $\quad j_a \leftarrow \tilde{\jmath}$	
$8a$ $\quad x_a \leftarrow x_{\tilde{\jmath}}$	
$\vdots$ $\quad \vdots$	
$10a$ $\quad j_a.tag.\mathrm{CAS}(static, a)$	
$\vdots$ $\quad \vdots$	
$13a$ $\quad$ **if** $x_a = j_a.x$ **:**	
$\vdots$ $\quad \vdots$	
	$\vdots$ $\quad \vdots$
	$16b$ $\quad i_b.x \leftarrow i_b.x + \tau_b$
	$17b$ $\quad i_b.tag \leftarrow static$
	$\vdots$ $\quad \vdots$

The identification of the target sphere j_a on thread a (statements $7a$ and $8a$) would be compromised if, before locking through the CAS statement at $10a$, it was changed in thread b, where the same sphere i_b is active (see statements $16b$, $17b$). However, the statement $13a$ checks that sphere j_a has not moved. If this condition is not satisfied, the thread a will be restarted (through statement $20a$). (See also Remark 12.)

2.4. Multithreaded ECMC (C++, OpenMP implementation)

Algorithm 6, discussed in this section, translates Algorithm 5 into C++ (OpenMP). The CAS statement and lock-free programming assure its efficiency. A sphere's attributes are again its position, its local time, and its tag. The latter is an atomic variable. We refer to line numbers in Algorithm 5.

Algorithm 6 (*Multithreaded ECMC (C++, OpenMP)*)**.** With initial values as in Algorithm 1, thread management is handled by OpenMP. The number of threads can be smaller than the number of active spheres. The multithreading stage transliterates the one of Algorithm 5. Statement 2ι of Algorithm 5 is implemented through a constraint graph (see Section 3.1). Statements 10ι through 13ι are expressed as follows in `MultiThreadECMC.cc`:

```
10ι →  j->tag.compare_exchange_strong(...static,...
11ι →  if (j->tag.load(memory_order) == iota)
12ι →  if (tau < distance)
13ι →  if (x == j->x),
```

where the `memory_order` qualifier may take on different values (see Section 3.2). Important differences with Algorithm 5 are discussed in Remarks 11 and 12. Output is as for Algorithm 1.

Code availability. *Algorithm 6 is implemented in* `MultiThreadECMC.cc`*. It is executed in several validation and benchmarking scripts (see Section 3.2).*

Remark 11 (*Active-sphere Necklaces*)**.** Algorithm 5 restarts thread ι if the target sphere j (for an active sphere i on the thread) is itself active on another thread. With periodic boundary conditions, active-sphere necklaces, where all target spheres are active, can deadlock the algorithm. To avoid this, Algorithm 6 moves sphere i up to contact with j before restarting (this is also used in Algorithm 4).

The source code of Algorithm 6 essentially translates that of Algorithm 5. The compiler may however change the order of execution for some statements in order to gain efficiency. (The memory access in modern multi-core processors can also be very complex and, in particular, thread-dependent.) Attributes, such as the `memory_order` qualifier in the CAS statement, may constrain the allowed changes of order. The reordering directives adopted in Algorithm 6 were chosen and validated with the help of extensive runs from randomly generated configurations. However, subtle pitfalls escaping notice through such testing can be exposed by explicitly reordering statements in Algorithm 5.

Remark 12 (*Memory-order Directives in Algorithms 5 and 6*)**.** In the `SequentialC5` test suite with $N = 5$, interchanging statements 15ι and 16ι in Algorithm 5 yields a spurious absorbing state, and invalidates the algorithm. The same test suite can also be input into Algorithm 6, where it passes the `Ordering.sh` validation test, even if the statements in `MultiCPP.cc` corresponding to 15ι and 16ι are exchanged. However, a 1 µs `pause` statement introduced in the C++ program between what corresponds to the (interchanged) statements 15ι and 16ι produces a $\sim 1\%$ error rate, illustrating that Algorithm 6 is unsafe without a protection of the order of the said statements. Safety may be increased through atomic position and local time variables, allowing the use of the `fetch_add()` operation to displace spheres.

3. Tools, validation protocols, benchmarks, and extensions

We now discuss the implementations of the algorithms of Section 2, as well as their validation protocols, benchmarks, and possible extensions. We also discuss the prospects of this method beyond this paper's focus on the interval between two breakpoints h and h'.

In our implementation of Algorithms 2, 4, and 6, a directed constraint graph encodes the possible pairs of active and target spheres as arrows $[i \to j]$ (see Section 3.1). The outdegree of this graph is at most three, and a rough constraint graph $\mathcal{G}^{(3)}$ with, usually, outdegree three for all vertices is easily generated. $\mathcal{G}^{(3)}$ may contain redundant arrows that cannot correspond to liftings. Our pruning algorithm eliminates many of them. We also prove that $\mathcal{G}^{\min}$, the minimal constraint graph, is planar. This may be of importance if disjoint parts of the constraint graph are stored on different CPUs, each with a number of dedicated threads. In general, we expect constraint graphs to be a useful tool for hard-sphere production codes, with typically $\mathcal{O}(N)$ liftings between changes of $\mathbf{v}$.

Validation scripts are discussed in Section 3.2. Scripts check that the liftings of standard cell-based ECMC are all accounted for in the used constraint graph. For the ECMC algorithms of Section 2, the set $\mathcal{L}$ of liftings provides the complete history of each run, and scripts check that they correspond to $\mathcal{L}^{\mathrm{ref}}$.

In Section 3.3, we benchmark Algorithm 6 and demonstrate a speed-up by an order of magnitude for a single CPU with 40 threads on an x86 CPU (see Section 3.3). The overhead introduced by multithreading ($\sim$ 2.4) is very reasonable. We then discuss possible extension of our methods (see Section 3.4).

3.1. Constraint graphs

For a given initial condition $\mathcal{C}_h$ and velocity $\mathbf{v}$, arrows $[i \to j]$ of the constraint graph $\mathcal{G}_{\mathbf{v}}$ represent possible liftings $l_t = ([i \to j], (\mathbf{x}, \mathbf{x}'), t)$ [26]. Arrows remain unchanged between breakpoints because spheres i and j with a perpendicular distance of less than 2σ cannot hop over one another (this argument can be adapted to periodic boundary conditions), and pairs with larger perpendicular distance are absent from $\mathcal{G}$. All constraint graphs $\mathcal{G}_{\mathbf{v}}$ are supersets of a minimal constraint graph $\mathcal{G}^{\min}_{\mathbf{v}} \equiv \mathcal{G}^{\min}_{-\mathbf{v}}$ (where the equivalence is understood as $[i \to j]_{\mathbf{v}} \equiv [j \to i]_{-\mathbf{v}}$).

Remark 13 (*Constraint Graphs and Convex Polytopes*)**.** Each arrow $[i \to j]$ of the constraint graph $\mathcal{G}_{\mathbf{v}}$ provides (for $\mathbf{v} = (1, 0)$) an inequality

$$x_i \le x_j - b_{ij} \tag{3}$$

that is tight ($x_i = x_j - b_{ij}$) when i lifts to j at contact (if there are configurations $\mathcal{C}$ where it is tight, then $[i \to j]$ belongs to $\mathcal{G}^{\min}_{\mathbf{v}}$). The set of inequalities defines a convex polytope. With periodic boundary conditions (unaccounted for in Eq. (3)), this polytope is

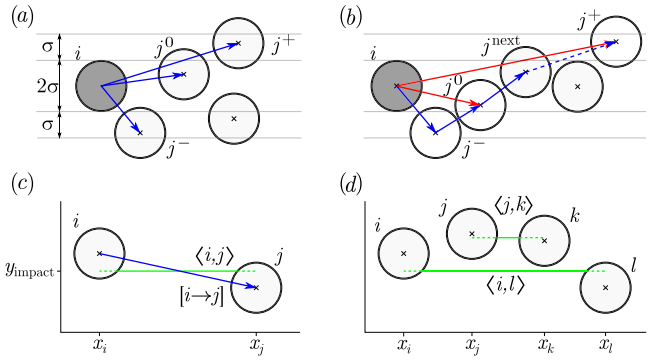

Fig. 5. Constraint graphs, pruning, and planarity. (a) Corridors of an active sphere i, with arrows $[i{\to}j^-]$, $[i{\to}j^0]$, and $[i{\to}j^+]$ belonging to $\mathcal{G}^{(3)}_{(1,0)}$. (b) Pruning of an arrow $[i{\to}j^+]$ through a sphere j^{next} without there being an arrow $[i{\to}j^{\text{next}}]$. (c) Spheres i and j, arrow $[i{\to}j]$, and impact path $\langle i,j\rangle$. Other spheres cannot cover $\langle i,j\rangle$. (d) Spheres i,j,k,l with $x_i<\cdots<x_l$ and impact paths $\langle i,l\rangle$ and $\langle j,k\rangle$.

infinite in the direction corresponding to uniform translation of all spheres with $\mathbf{v}$ (see [26]).

Remark 14 (*Constraint Graphs and Irreducibility*)**.** Rigorously, we define the constraint graph $\mathcal{G}^{\min}_{\mathbf{v}}$ as the set of arrows $[i\to j]$ that are encountered from $\mathcal{C}_h$ by Algorithm 1 (or, equivalently, Algorithm 2) at liftings l_t $\forall t\in(-\infty,\infty)$. The liftings for $t<0$ can be constructed because of time-reversal invariance (see Remark 3). For the same reason, we have $\mathcal{G}^{\min}_{\mathbf{v}}\equiv\mathcal{G}^{\min}_{-\mathbf{v}}$, and the set of arrows reached from $\mathcal{C}_h$ is equivalent to that reached from any configuration that is reached from $\mathcal{C}$ (and in particular $\mathcal{C}_{h'}$). While we expect ECMC to be irreducible in the polytope defined through the inequalities in Eq. (3), we do not require irreducibility for the definition of $\mathcal{G}^{\min}$.

Between breakpoints, the active sphere i can lift to at most three other spheres, namely the sphere j^0 minimizing the time of flight τ_{ij} in a corridor of width 2σ around the center of i, and likewise the closest-by sphere j^+ in the corridors $[\sigma,2\sigma]$ and sphere j^- in the corridor $[-2\sigma,-\sigma]$ (see Fig. 5a). The set of arrows $\{[i\to j^0],[i\to j^-],[i\to j^+]\ \forall i\in\{1,\dots,N\}\}$ constitutes the constraint graph $\mathcal{G}^{(3)}_{\mathbf{v}}$, which is thus easily computed. Except for small systems (where the corridors may be empty), $\mathcal{G}^{(3)}$ has outdegree three for all spheres i. However, its indegree is not fixed. The constraint graph $\mathcal{G}^{(3)}$ is not necessarily locally planar,[4] and in the embedding provided by the sphere centers of a given configuration, non-local arrows can be present (see Fig. 6a). However, $\mathcal{G}^{\min}$ can be proven to be locally planar (see Fig. 6b).

Lemma 4. *The graph $\mathcal{G}^{\min}$ is locally planar, and any sphere configuration that can be reached between breakpoints provides a locally planar embedding.*

Proof. We first consider two spheres i and j for $\mathbf{v}=(1,0)$ in the plane (without taking into account periodic boundary conditions). The arrow $[i\to j]$ is drawn by connecting the centers of i and j. The impact path $\langle i,j\rangle$ is the horizontal line segment connecting (x_i,y^{impact}) and (x_j,y^{impact}) where y^{impact} is the vertical position at which the two spheres can touch by moving them with $\mathbf{v}$ (see Fig. 5c). If the arrow $[i\to j]$ exists, no other sphere can intersect the impact path $\langle i,j\rangle$.

[4] "Locally planar" means that any subgraph that does not sense the periodic boundary conditions is planar.

For four spheres i,j,k,l, we now show that no two arrows between spheres can cross each other. The x-values can be ordered as $x_i<x_j<x_k<x_l$ (again without taking into account periodic boundary conditions). Two arrows between three spheres trivially cannot cross. For arrows between two pairs of spheres, arrows $[i{\to}j]$ and $[k{\to}l]$ cannot cross. Likewise, if there is an arrow $[i{\to}k]$, then sphere j must be on one side of the impact path $\langle i,k\rangle$, and k must be on the other side of $\langle j,l\rangle$, so that arrows $[i{\to}k]$ and $[j{\to}l]$ cannot cross. Finally, if arrow $[i{\to}l]$ exists, then j and k must be on the same side of the impact path $\langle i,l\rangle$ in order to have an impact path. But then, $[j{\to}k]$ cannot cross $[i{\to}l]$ (see Fig. 5d). □

The minimal constraint graph $\mathcal{G}^{\min}$ is more difficult to compute than $\mathcal{G}^{(3)}$ because the underlying "redundancy detection" problem is not strictly polynomial in system size, although practical algorithms exist [27]. However, $\mathcal{G}^{(3)}$ can be pruned of redundant constraints that correspond to pairs of spheres i and j that are prevented from lifting by other spheres. For example, given arrows $[i\to j]$, $[j\to k]$ and $[i\to k]$, the latter can be "first-order" pruned (eliminated with one intermediary, namely j) if $b_{ij}+b_{jk}>b_{ik}$ (in Fig. 5, $[i{\to}j^+]$ can be pruned for this reason). The presence of the arrow $[j\to k]$ is not necessary to make this argument work (see Fig. 5b). Pruning can be taken to higher orders. To second order, if $b_{ij}+b_{jk}+b_{kl}>b_{il}$, then the arrow $[i{\to}l]$ can be eliminated. Finally, any arrow $[i{\to}j]$ in $\mathcal{G}_{\mathbf{v}}$ can be pruned through symmetrization if it is unmatched by $[j{\to}i]$ in $\mathcal{G}_{-\mathbf{v}}$ because $\mathcal{G}^{\min}_{\mathbf{v}}\equiv\mathcal{G}^{\min}_{-\mathbf{v}}$, with $\mathcal{G}[\mathbf{v}]$ and $\mathcal{G}[-\mathbf{v}]$ obtained separately (see Remark 3).

Rarely, arrows can be eliminated by symmetrizing graphs that were pruned to third or fourth order, and constraint graphs that are obtained in this way appear close to $\mathcal{G}^{\min}$ (see Section 3.2).

Code availability. *The constraint graph $\mathcal{G}^{(3)}$ is constructed in* `GenerateG3.py` *and pruned to $\mathcal{G}$ in* `PruneG.py`*. The program* `GraphValidateCellECMC.cc` *runs cell-based ECMC to verify the consistency of $\mathcal{G}$.*

3.2. Validation

Our programs apply to arbitrary density $\eta=N\pi\sigma^2/L^2$ and linear size L of the periodic square box (with $N=M^2$). We provide sets of configurations and constraint graphs for validation and benchmarking. One such set consists in a configuration $\mathcal{C}^{256}$ at $M=256$, and for $\eta=0.708$ and a fourth-order symmetrized constraint graph $\mathcal{G}^{256}$. Where applicable, the number of active spheres varies as $k=1,2,4,\dots,k^{\max}$ and the number of threads as $n_t=1,2,3,\dots,n_t^{\max}$. For fixed k and n_t, there are n_{run} runs that vary h and h'.

Constraint-graph validation. Constraint graphs are generated in the `Setup.sh` script. The `GraphValidateCellECMC` test performs cell-based ECMC derived from `CellECMC.f90` [3,23], where spheres are assigned to local cells and neighborhood-cell searches identify possible liftings. Cell-based ECMC must exclusively solicit liftings accounted for in $\mathcal{G}$. The `GraphValidateCellECMC` test also records the sweep (lifting per sphere) at which an arrow $[i{\to}j]\in\mathcal{G}$ is first solicited in a lifting and compares the time evolution of the average number of solicited arrows with its average outdegree. The $\mathcal{G}^{256}$ constraint graph passes the validation test with $t=2\times10^7$ sweeps. The outdegree of $\mathcal{G}^{256}$ is 2.1, and 98.7% of its arrows are solicited during the test. Logarithmic extrapolation (with $1/(\ln t)^\alpha$, $\alpha=1.7$) suggests that $\mathcal{G}^{256}$ essentially agrees with $\mathcal{G}^{\min}$ (see Fig. 6c). Use of $\mathcal{G}^{256}$ rather than $\mathcal{G}^{(3)}$ speeds up ECMC, but further performance gains through additional pruning are certainly extremely limited.

Validation of Algorithms 4 and 6. Our implementations of Algorithms 4 and 6 are modified as discussed in Remark 7. Runs

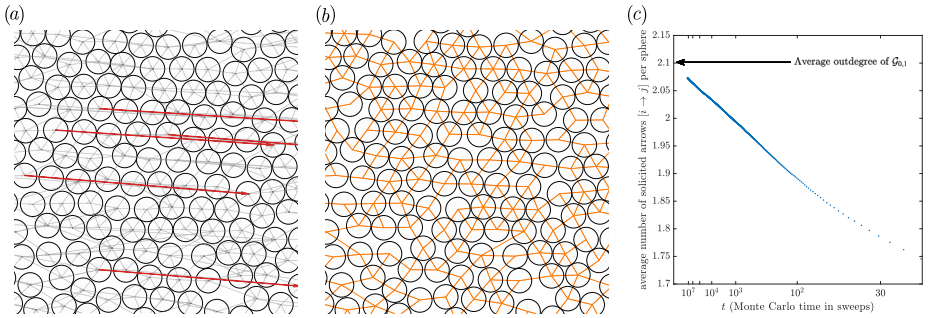

Fig. 6. ECMC constraint graphs for $\mathcal{C}^{256}$ (see Section 3.2 for definition). (a) $\mathcal{G}^{(3)}$ for this configuration (detail), with highlighted non-local arrows. (b) $\mathcal{G}^{256}$ (same detail), obtained from $\mathcal{G}^{(3)}$ through fourth-order pruning followed by symmetrization. (c) Number of solicited arrows in $\mathcal{G}^{256}$ in a long cell-based ECMC run, compared to its average outdegree.

compute the set $\mathcal{L}_{t^*}$ to the earliest horizon-violation time t^* (with $t^* = h'$ if the run concludes successfully). The `PValidateECMC.sh` test first advances $\mathcal{C}^{256} = \mathcal{C}_{t=0}$ to a random breakpoint h (using $\mathcal{L}^{\text{ref}}$). Each test run is in the interval $[h, h']$, where h' is randomly chosen. To pass the validation test, $\mathcal{L}_{t^*}$ must for each run agree with $\mathcal{L}^{\text{ref}}$ (see Section 4 for details of scripts used). Algorithm 4 passes the `PValidateECMC.sh` test with $n_{\text{run}} = 1 \times 10^3$ for $k^{\max} = 8192$.

Our x86 computer has two Xeon Gold 6230 CPUs with variable frequency from 2.1 GHz to 3.9 GHz, each with 20 cores and 40 hardware threads. We use OpenMP directives to restrict all threads to a single CPU. We consider again $\mathcal{C}^{256} = \mathcal{C}_{t=0}$ as the initial configuration, and then run the program from h to h'. On our x86 CPU, Algorithm 6 passes the `CValidateECMC.sh` test with $n_{\text{run}} = 1 \times 10^3$, $k^{\max} = 8192$ and $n_\iota^{\max} = 40$.

On our ARM CPU (Nvidia Jetson with Cortex A57 CPU (at 1.43 GHz) with four cores and four hardware threads), we again consider $\mathcal{C}^{256} = \mathcal{C}_{t=0}$ as initial configuration. For the same system parameters as above, Algorithm 6 passes the `CValidateECMC.sh` test with $n_{\text{run}} = 1 \times 10^3$, $k^{\max} = 8192$ and $n_\iota^{\max} = 4$. The ARM architecture allows dynamic re-ordering of operations, and the separate validation test more severely scrutinizes thread interactions than for the x86 CPU.

On both CPUs, Algorithm 6 passes the `CValidateECMC.sh` test with the following choices of `memory_order` directives:

`memory_order_relaxed`**.** This most permissive memory ordering of the C++ memory model imposes no constraints on compiler optimization or dynamic re-ordering of operations by the processors, and only guarantees the atomic nature of the CAS operation. Such re-orderings are more aggressive on ARM CPUs than on x86 CPUs. This memory ordering does not guarantee that the statements constituting the lock-less lock are executed as required (see Remark 12).

`memory_order_seq_cst` for all memory operations on the tag attribute. This directive imposes the sequential-consistency model (see Remark 12) for each access of the tag attribute. It slows down the code by 40% compared to the `memory_order_relaxed` directive.

`memory_order_acquire` on `load`, `memory_order_release` on `store`. This directive implies "acquire–release" semantics on the tag attribute. It imposes a lock-free exchange at each operation on the tag attribute, so that all variables, including positions and local times, are synchronized between threads during tag access. CAS remains `memory_order_seq_cst`. This directive maintains speed compared to `memory_order_relaxed`, yet provides better guarantees on the propagation of variable modification between threads. `MultiThreadECMC.cc` compiles by default with this directive.

3.3. Benchmarks for Algorithm 6 (x86 and ARM)

Algorithm 6 is modified as discussed in Remark 7 (program execution continues in spite of horizon violations) and used for large values of h'. This measures the net cost of steady-state thread interaction, without taking into account thread-setup times. We report here on results of the `BenchmarkECMC.sh` script for $\mathcal{C}^{256}$ as an initial configuration and $\mathcal{G}^{256}$ with $k = 40$ active spheres. The number of threads varies as $n_\iota = 1, 2, 3, \ldots, n_\iota^{\max}$, with $n_{\text{run}} = 20$.

On our x86 CPU (see Section 3.2), the `BenchmarkECMC.sh` script is parametrized with $n_\iota^{\max} = 40$. The benchmark speed increases roughly linearly up to 20 threads (reaching a speed-up of 10 for 20 threads), and then keeps improving more slowly with a maximum for 40 threads at a speed-up of 14 and an absolute speed of $\sim 1.6 \times 10^{12}$ events/hour (see Fig. 7). The variable frequency of Xeon processors under high load may contribute to this complex behavior. On a single thread, our program runs 2.2 times slower than an unthreaded code, due to the eliminated overhead from threading constructs. The original `CellECMC.f90` cell-based production code generates 3×10^{10} events/hour. The use of a constraint graph, rather than a cell-based search, thus improves performance by almost an order of magnitude, if the set-up of $\mathcal{G}$ is not accounted for.

On our ARM CPU, the `BenchmarkECMC.sh` script is parametrized with $n_\iota^{\max} = 4$. The benchmark speed increases as the number of threads, reaching a speed-up of 3.8 for $n_\iota = 4$. The absolute speed is about six times smaller than for our x86 CPU for a comparable number of threads, as may be expected for a low-power processor designed for use in mobile phones.

3.4. Birthday problem, full ECMC, multi-CPU extensions

In this section, we treat some practical aspects for the use of Algorithm 6.

Birthday problem. We analyze multithreaded ECMC in terms of the (generalized) birthday problem, which considers the probability p that two among k' integers (modeling individuals) sampled from a discrete uniform distribution in the set $\{1, 2, \ldots, N'\}$ (modeling birthdays) are the same. For large N', $p \sim [1 - \exp(-k'^2/(2N'))]$ [28], which is small if $k' \lesssim \sqrt{N'}$. At constant density η, sphere radius σ, velocity $\mathbf{v}$, and time interval $h' - h$, each active chain ι is almost restricted to a region of constant area, whereas the total area of the simulation box is ηN. We may

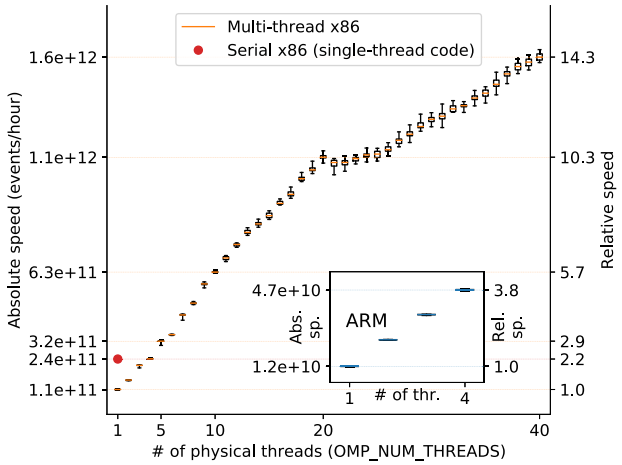

Fig. 7. Output of the `BenchmarkECMC.sh` script for Algorithm 6 (five-number summary of 20 runs) for $k = 80$ on an x86 CPU with 20 cores and $1, \ldots, 40$ threads, and for serial code that processes active chains sequentially. Inset: Output of the script for a four-core ARM CPU.

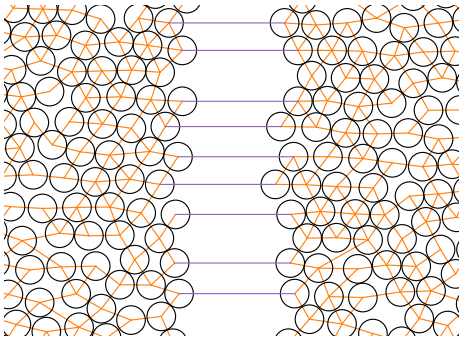

Fig. 8. A constraint graph doubled up for a multi-CPU implementation of Algorithm 6 (detail of $\mathcal{C}^{256}$ configuration shown). Interface vertices appear on both sides.

suppose that the k active spheres are randomly positioned in the simulation box broken up into a grid of $\propto N$ constant-area cells. For $k \lesssim \sqrt{N}$, we expect the probability that one of these cells contains two active spheres to remain constant for $N \to \infty$, and therefore also the probability of an update-order violation for constant $h' - h$.

Restarts. Our algorithms reproduce output of Algorithm 2 only if they do not abort. In production code, the effects of horizon violations will have to be repaired. Two strategies appear feasible. First, the algorithm may restart the run from a copy of $\{\mathcal{C}_h, \mathcal{A}_h\}$ at the initial breakpoint h, and choose a smaller breakpoint h'', for example the time of abort. The successful termination of this restart is not guaranteed, as the individual threads may organize differently. Second, the time evolution may be reconstructed from $\mathcal{L}_{t^*}$ to the earliest horizon-violation time t^* (see Remark 7), and t^* may then be used as the subsequent initial breakpoint. Besides an efficient restart strategy, a multithreaded production code will also need an efficient parallel algorithm for computing $\mathcal{G}$ after a change of **v**.

Multi-CPU implementations. Algorithm 6 is spelled out for a single shared-memory CPU and for threads that may access attributes of all spheres (see statement 10ι in Algorithm 5 and Remark 11). However, thread interactions are local and immutable in between breakpoints (as evidenced by the constraint graphs). This invites generalizations of the algorithm to multiple CPUs (each of them with many threads). Most simply, two CPUs could administer disjoint parts of the constraint graph, for example with interface vertices doubled up on both of them (see Fig. 8). In this way, an active sphere arriving at an interface would simply be copied out to the neighboring CPU. The generalization to multiple CPUs appears straightforward.

4. Available computer code

All implemented algorithms and used scripts that are made available on GitHub in `ParaSpheres`, a public repository which is part of a public GitHub organization.[5] Code is made available under the GNU GPLv3 license (for details see the `LICENSE` file).

The repository can be forked (that is, copied to an outside user's own public repository) and from there studied, modified and run in the user's local environment. Users may contribute to the `ParaSpheres` project *via* pull requests (see the `README.md` and `CONTRIBUTING.md` files for instructions and guidelines). All communication (bug reports, suggestions) take place through GitHub "Issues", that can be opened in the repository by any user or contributor, and that are classified in GitHub projects.

Implemented algorithms. The following programs are located in the directory tree under their language (`F90`, `Python` or `CPP`) and in similarly named subdirectories, that all contain `README` files for further details. Some of the longer programs are split into modules.

Code/Directory	Algorithm/Usage
`CellECMC.f90`	Cell-based production ECMC [3]
`GenerateG3.py`	Generate $\mathcal{G}^{(3)}$ (Section 3.1)
`PruneG.py`	Prune $\mathcal{G}$ (Section 3.1)
`GraphValidateCellECMC.cc`	Validate $\mathcal{G}$ against cell-based ECMC
`GlobalTimeECMC.py`	Algorithm 2 (Section 2.1)
`SingleThreadLocalTimeECMC.py`	Algorithm 4 (Section 2.2)
`SequentialMultiThreadECMC.py`	Algorithm 5 (Section 2.3)
`MultiThreadECMC.cc`	Algorithm 6 (Section 2.4)

Scripts and validation suites. The `Scripts` directory provides the following bash scripts to compile and run groups of programs and to reproduce all our results:

Script	Summary of usage
`Setup.sh`	Prepare $\mathcal{C}_{t=0}$, $\mathcal{G}$, $\mathcal{L}^{\text{ref}}$
`SequentialC4.sh`	Test suite for Algorithm 5 with $N = 4$
`SequentialC5.sh`	Test suite for Algorithm 5 with $N = 5$ (see Remark 12)
`Ordering.sh`	Test suite for Algorithm 6 with $N = 5$ (see Remark 12)
`ValidateG.sh`	Validate constraint graph
`PValidateECMC.sh`	Validate Algorithm 4 against $\mathcal{L}^{\text{ref}}$
`CValidateECMC.sh`	Validate Algorithm 6 against $\mathcal{L}^{\text{ref}}$
`BenchmarkECMC.sh`	Benchmark `MultiThreadECMC.cc`, generate Fig. 7

In the `Setup.sh` script, `CellECMC.f90` first produces a sample $\mathcal{C}_0$ such that the unidirectional dynamics in $\mathcal{C}_0$ is practically aperiodic. It then generates $\mathcal{G}^{(3)}$ with `GenerateG3.py`,

[5] The organization's url is https://github.com/jellyfysh.

and runs `PruneG.py` to output $\mathcal{G}$. Finally, it runs `GlobalTimeECMC.py` for each set $\mathcal{A}_0$, in order to generate several $\mathcal{L}^{\text{ref}}$. The `ValidateG.sh` script uses `GraphValidateCellECMC.cc` to run cell-based ECMC, and verifies that all liftings are accounted for in $\mathcal{G}$. It also tracks the solicitation of arrows as a function of time. The `PValidateECMC.sh` script validates `SingleThreadLocalTimeECMC.py` by comparing the sets of liftings with $\mathcal{L}^{\text{ref}}$ from `Setup.sh`. The `CValidateECMC.sh` script does the same for `MultiThreadECMC.cc`. `BenchmarkECMC.sh` benchmarks `MultiThreadECMC.cc` for different numbers of threads. The test suites are concerned with small-N configurations.

5. Conclusions and outlook

In this paper, we presented an event-driven multithreaded ECMC algorithm for hard spheres which enforces thread synchronization at infrequent breakpoints only. Between breakpoints, spheres carry and update local times. Possible inconsistencies are locally detected through a horizon condition. Within ECMC, our method avoids the scheduling problem that has historically plagued event-driven molecular dynamics. This is possible because in ECMC only few spheres move at any moment, and all have the same velocity. Conflicts are thus exceptional, and little information is exchanged between threads. We relied on the generalized birthday problem to show that our algorithm remains viable up to a number of threads that grows as the square root of the number of spheres, a setting relevant for the simulation of millions of spheres for modern commodity servers with $\sim$100 threads. The mapping of Algorithm 5 onto an absorbing Markov chain allowed us to prove its correctness (for a given lifted initial configuration) and to rigorously analyze side effects of code re-orderings in the multithreaded C++ code.

Our algorithm is presently implemented between two global breakpoint times, where it achieves considerable speed-up with respect to sequential ECMC. A fully practical multithreaded ECMC code that greatly outperforms cell-based algorithms appears within reach. It is still a challenge to understand whether multithreaded ECMC applies to general interacting-particle systems.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

W.K. acknowledges support from the Alexander von Humboldt Foundation, Germany. We thank E. P. Bernard for allowing his original hard-sphere ECMC production code to be made available.

References

[1] E.P. Bernard, W. Krauth, D.B. Wilson, Phys. Rev. E 80 (2009) 056704, http://dx.doi.org/10.1103/PhysRevE.80.056704, URL http://link.aps.org/doi/10.1103/PhysRevE.80.056704.
[2] M. Michel, S.C. Kapfer, W. Krauth, J. Chem. Phys. 140 (5) (2014) 054116, http://dx.doi.org/10.1063/1.4863991, arXiv:1309.7748.
[3] E.P. Bernard, W. Krauth, Phys. Rev. Lett. 107 (2011) 155704, http://dx.doi.org/10.1103/PhysRevLett.107.155704, URL http://link.aps.org/doi/10.1103/PhysRevLett.107.155704.
[4] S.C. Kapfer, W. Krauth, Phys. Rev. Lett. 114 (2015) 035702, http://dx.doi.org/10.1103/PhysRevLett.114.035702, URL http://link.aps.org/doi/10.1103/PhysRevLett.114.035702.
[5] M. Hasenbusch, S. Schaefer, Phys. Rev. D 98 (2018) 054502, http://dx.doi.org/10.1103/PhysRevD.98.054502, URL https://link.aps.org/doi/10.1103/PhysRevD.98.054502.
[6] E.P. Bernard, W. Krauth, Addendum to Event-chain Monte Carlo algorithms for hard-sphere systems, Phys. Rev. E 86 (1) http://dx.doi.org/10.1103/physreve.86.017701.
[7] S.C. Kapfer, W. Krauth, Phys. Rev. Lett. 119 (2017) 240603, http://dx.doi.org/10.1103/PhysRevLett.119.240603, URL https://link.aps.org/doi/10.1103/PhysRevLett.119.240603.
[8] M.F. Faulkner, L. Qin, A.C. Maggs, W. Krauth, J. Chem. Phys. 149 (6) (2018) 064113, http://dx.doi.org/10.1063/1.5036638.
[9] J. Harland, M. Michel, T.A. Kampmann, J. Kierfeld, Europhys. Lett. 117 (3) (2017) 30001, URL http://stacks.iop.org/0295-5075/117/i=3/a=30001.
[10] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller, J. Chem. Phys. 21 (1953) 1087–1092, http://dx.doi.org/10.1063/1.1699114.
[11] B.J. Alder, T.E. Wainwright, J. Chem. Phys. 27 (1957) 1208–1209, http://dx.doi.org/10.1063/1.1743957.
[12] B.J. Alder, T.E. Wainwright, I. General Method, J. Chem. Phys. 31 (1959) 459–466, http://dx.doi.org/10.1063/1.1730376.
[13] D.C. Rapaport, J. Comput. Phys. 34 (1980) 184–201, http://dx.doi.org/10.1016/0021-9991(80)90104-7.
[14] M. Isobe, Mol. Simul. 42 (16) (2016) 1317–1329, http://dx.doi.org/10.1080/08927022.2016.1139106.
[15] B.D. Lubachevsky, Int. J. Comput. Simul. 2 (1992) 373–411.
[16] B. Lubachevsky, ACM SIGSIM Simul. Digest 23 (1993) 60–67, http://dx.doi.org/10.1145/174134.158467.
[17] A.G. Greenberg, B.D. Lubachevsky, I. Mitrani, ACM Trans. Model. Comput. Simul. (TOMACS) 6 (2) (1996) 107–136.
[18] A.T. Krantz, ACM Trans. Model. Comput. Simul. 6 (3) (1996) 185–209, http://dx.doi.org/10.1145/235025.235030.
[19] M. Marin, Proceedings 11th Workshop on Parallel and Distributed Simulation, 1997, pp. 164–171, http://dx.doi.org/10.1109/PADS.1997.594602.
[20] S. Miller, S. Luding, J. Comput. Phys. 193 (1) (2004) 306–316, http://dx.doi.org/10.1016/j.jcp.2003.08.009, arXiv:physics/0302002.
[21] P. Diaconis, S. Holmes, R.M. Neal, Ann. Appl. Probab. 10 (2000) 726–752.
[22] D.A. Levin, Y. Peres, E.L. Wilmer, Markov Chains and Mixing Times, American Mathematical Society, 2008.
[23] M. Engel, J.A. Anderson, S.C. Glotzer, M. Isobe, E.P. Bernard, W. Krauth, Phys. Rev. E 87 (2013) 042134, http://dx.doi.org/10.1103/PhysRevE.87.042134, URL http://link.aps.org/doi/10.1103/PhysRevE.87.042134.
[24] Lamport, IEEE Trans. Comput. C-28 (9) (1979) 690–691, http://dx.doi.org/10.1109/tc.1979.1675439.
[25] H.J. Boehm, L. Crowl, C++ atomic types and operations, 2009, URL http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2007/n2427.html.
[26] S.C. Kapfer, W. Krauth, J. Phys. Conf. Ser. 454 (1) (2013) 012031, http://dx.doi.org/10.1088/1742-6596/454/1/012031, URL http://stacks.iop.org/1742-6596/454/i=1/a=012031.
[27] K. Fukuda, B. Gärtner, M. Szedlák, Ann. Oper. Res. 265 (1) (2016) 47–65, http://dx.doi.org/10.1007/s10479-016-2385-z.
[28] F.H. Mathis, SIAM Rev. 33 (2) (1991) 265–270, http://dx.doi.org/10.1137/1033051.

Publication 2: Sparse hard-disk packings and local Markov chains

Journal of Statistical Physics (2022) 187:31
https://doi.org/10.1007/s10955-022-02908-4

Sparse Hard-Disk Packings and Local Markov Chains

Philipp Höllmer[1] · Nicolas Noirault[2] · Botao Li[2] · A. C. Maggs[3] · Werner Krauth[2]

Received: 8 October 2021 / Accepted: 14 March 2022 / Published online: 22 April 2022

Abstract

We propose locally stable sparse hard-disk packings, as introduced by Böröczky, as a model for the analysis and benchmarking of Markov-chain Monte Carlo (MCMC) algorithms. We first generate such Böröczky packings in a square box with periodic boundary conditions and analyze their properties. We then study how local MCMC algorithms, namely the Metropolis algorithm and several versions of event-chain Monte Carlo (ECMC), escape from configurations that are obtained from the packings by slightly reducing all disk radii by a relaxation parameter. We obtain two classes of ECMC, one in which the escape time varies algebraically with the relaxation parameter (as for the local Metropolis algorithm) and another in which the escape time scales as the logarithm of the relaxation parameter. A scaling analysis is confirmed by simulation results. We discuss the connectivity of the hard-disk sample space, the ergodicity of local MCMC algorithms, as well as the meaning of packings in the context of the NPT ensemble. Our work is accompanied by open-source, arbitrary-precision soft-

Communicated by Ludovic Berthier.

Philipp Höllmer acknowledges support from the Studienstiftung des deutschen Volkes and from Institut Philippe Meyer. Werner Krauth acknowledges support from the Alexander von Humboldt Foundation.

Philipp Höllmer
hoellmer@physik.uni-bonn.de

Nicolas Noirault
nicolas.noirault@laposte.net

Botao Li
botao.li@phys.ens.fr

A. C. Maggs
anthony.maggs@espci.fr

Werner Krauth
werner.krauth@ens.fr

[1] Physikalisches Institut and Bethe Center for Theoretical Physics, University of Bonn, Nussallee 12, 53115 Bonn, Germany

[2] Laboratoire de Physique de l'Ecole normale supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université de Paris, Paris, France

[3] CNRS Gulliver, ESPCI Paris, Université PSL, 10 rue Vauquelin, 75005 Paris, France

ware for Böröczky packings (in Python) and for straight, reflective, forward, and Newtonian ECMC (in Go).

Keywords Hard-disk packings · Stability · Markov chains · Hard-disk model · Event-chain Monte Carlo · Mixing times

1 Introduction

The hard-disk system is a fundamental statistical-physics model that has been intensely studied since 1953. Numerical simulations, notably Markov-chain Monte Carlo [1] (MCMC) and event-driven molecular dynamics [2], have played a particular role in its study. The existence of hard-disk phase transitions [3] was asserted as early as 1962. The recent identification of the actual transition scenario [4] required the use of a modern event-chain Monte Carlo (ECMC) algorithm [5, 6].

The hard-disk model has been much studied in mathematics. Even today, the existence of a phase transition has not been proven [7, 8]. A fundamental rigorous result is that the densest packing of N equal hard disks (for $N \to \infty$) arranges them in a hexagonal lattice [9]. This densest packing is locally stable, which means that no single disk can move infinitesimally in the two-dimensional plane. The densest packing is furthermore collectively stable, which means that no subset of disks can move at once, except if the collective infinitesimal move corresponds to symmetries, as for example uniform translations in the presence of periodic boundary conditions [10–12]. In 1964, Böröczky [13] constructed two-dimensional disk packings that are sparse, that is, have vanishing density in the limit $N \to \infty$. The properties of these Böröczky packings are very different from those of the densest hexagonal lattice. Infinitesimal motion of just a single disk remains impossible, so that Böröczky packings are locally stable. However, coherent infinitesimal motion of more than one disk does allow escape from Böröczky packings so that they are not collectively stable.

In this work, we construct finite-N Böröczky packings in a fixed periodic box and use them to build initial configurations for local Markov-chain Monte Carlo (MCMC) algorithms, namely the reversible Metropolis algorithm [1, 14] and several variants [5, 15, 16] of non-reversible ECMC. In the Metropolis algorithm, single disks are moved one by one within a given range δ. A Böröczky packing traps the local Metropolis algorithm if δ is small enough, because all single-disk moves are rejected. ECMC is by definition local. It features individual infinitesimal displacements of single disks, and it also cannot escape from a Böröczky packing. We thus consider ε-relaxed Böröczky configurations that have the same disk positions as the Böröczky packings but with disk radii reduced by a factor $(1 - \varepsilon)$. Here, $\varepsilon \gtrsim 0$ is the relaxation parameter. Our scaling theory for the escape times from ε-relaxed Böröczky configurations predicts the existence of two classes of local Markov-chain algorithms. In one class, escape times grow as a power of the relaxation parameter ε, whereas the other class features only logarithmic growth. Numerical simulations confirm our scaling theory, whose power-law exponents we conjecture to be exact. The ε-relaxed Böröczky configurations are representative of a finite portion of sample space. For a fixed number of disks, the growth of the escape times thus leads to the existence of a small but finite fraction of sample space that cannot be escaped from or even accessed by local MCMC in a given upper limit of CPU time. More generally, we discuss the apparent paradox that the lacking proof for the connectedness of the hard-disk sample space, on the one hand, might render local MCMC non-irreducible (that is, "non-ergodic") but, on the other hand, does not invalidate their practical use. We resolve this paradox by considering the NPT ensemble (where the pressure is conserved instead of

the volume). We moreover advocate the usefulness of ε-relaxed Böröczky configurations for modeling bottlenecks in MCMC and consider the comparison of escape times from these configurations as an interesting benchmark. We provide open-source arbitrary-precision software for Böröczky packings and for ECMC. Several of the ECMC algorithms can evolve towards numerical gridlock, that can be diagnosed and studied using our arbitrary-precision software.

This work is organized as follows. In Sect. 2, we construct Böröczky packings following the original proposal [13] and a variant due to Kahle [17], and we analyze their properties. In Sect. 3, we discuss local MCMC algorithms and present analytical and numerical results for the escape times from the ε-relaxed Böröczky configurations. In Sect. 4, we analyze algorithms and their escape times and discuss fundamental aspects, among them irreducibility, statistical ensembles, as well as the question of bottlenecks, and the difference between local and non-local MCMC methods. In the conclusion (Sect. 5), we point to several extensions and place our findings into the wider context of equilibrium statistical mechanics, the physics of glasses and the mechanics of granular materials. In Appendix A, we present further numerical analysis and, in Appendix B, we introduce our open-source arbitrary-precision software package `BigBoro` for Böröczky packings and for ECMC.

2 Böröczky Packings

In the present section, we discuss Böröczky packings of N disks of radius $\sigma = 1$ in a periodic square box of sides L. The density η is the ratio of the disk areas to that of the box:

$$\eta = N\pi\sigma^2/L^2. \tag{1}$$

For concreteness, the central simulation box ranges from $-L/2$ to $L/2$ in both the x and the y direction. The periodic boundary conditions map the central simulation box onto an infinite hard-disk system with periodically repeated boxes or, equivalently, onto a torus. A Böröczky packing is locally stable, and each of its N disks is blocked—at a distance 2σ—by at least three other disks (taking into account periodic boundary conditions), with the contacts not all in the same half-plane. The opening angle of a disk i, the largest angle formed by the contacts to its neighbors, is then always smaller than π. The maximum opening angle is the largest of the N opening angle of all disks. Clearly, a locally stable packing cannot be escaped from through the infinitesimal single-disk moves of ECMC or, in Metropolis MCMC, through steps of small enough range. Only collective infinitesimal moves of all disks may escape from the packing.

In a nutshell, Böröczky packings (see Sect. 2.1 for their construction) consist in cores and branches (as visible in Fig. 1). The original Ref. [13] mainly focused on Böröczky packings in an infinite plane, but also sketched how to generalize the packings to the periodic case. Böröczky packings can exist for different cores, and they depend on a bounding curve (more precisely: a convex polygonal chain) which encloses the branches, and which can be chosen more or less freely (see Sect. 2.2 for the properties of Böröczky packings, including the collective infinitesimal escape modes from them).

2.1 Construction of Böröczky Packings

In the central simulation box, a finite-N Böröczky packing is built on a central core placed around $(0, 0)$ (see Sect. 2.1.1 for a discussion of cores). This core connects to four periodic copies of the core centered at $(L, 0)$, $(0, L)$, $(-L, 0)$, and $(0, -L)$ by branches that have k

separate layers (see Sects. 2.1.2 and 2.1.3 for a detailed discussion of branches). A Böröczky packing shares the symmetries of the central simulation box. Cores with different shapes, as for example that of a triangle, yield Böröczky packings in other geometries (see [13, 17] and [18, Sect. 9.3]).

2.1.1 Böröczky Core, Kahle Core

We consider Böröczky packings with two different cores, either the Böröczky core or the Kahle core. Both options are implemented in the `BigBoro` software package (see Appendix B). The Böröczky core [13] consists of 20 disks (see Fig. 1a). Using reflection symmetry about coordinate axes and diagonals, this core can be constructed from four disks at coordinates $(\sqrt{2}, 0)$, $(2+\sqrt{2}, 0)$, $(2+\sqrt{6}/2+1/\sqrt{2}, \sqrt{6}/2+1/\sqrt{2})$, and $(2+\sqrt{6}/2+1/\sqrt{2}, 2+\sqrt{6}/2+1/\sqrt{2})$ (see highlighted disks in Fig. 1a). The Kahle core [17], with a total of 8 disks, is constructed from two disks at coordinates $(1, 1)$, and $(1+\sqrt{3}, 0)$, using the same symmetries (see highlighted disks in Fig. 1b). The Böröczky core for $k = 0$, that is without the branches included in Fig. 1a, is only locally stable if repeated periodically in a central simulation box that fully encloses the core disks, with $L/2 = 3+\sqrt{6}/2+1/\sqrt{2}$. The Kahle core, again without branches, can be embedded in two non-equivalent ways into a periodic structure. When the outer-disk centers are placed on the cell boundaries, with $L/2 = 1+\sqrt{3}$, it forms a collectively stable packing with no remaining degrees of freedom other than uniform translations. Alternatively, it only forms a locally stable packing, with the possibility of non-trivial collective deformations, if the outer disks are enclosed in a larger simulation cell, with $L/2 = 2+\sqrt{3}$. These two cores are the seeds from which larger and less dense Böröczky packings are now constructed and studied.

2.1.2 Branches—Infinite-Layer Case (Infinite *N*)

Following Ref. [13], we first construct infinite branches ($k = \infty$) that correspond to the $N \to \infty$ and $\eta \to 0$ limits, without periodic boundary conditions. One such branch is attached to each of the four sides of the central core so that all disks are locally stable. The horizontal branch that extends from the central core in the positive x-direction is symmetric about the x-axis. The half branch for $y \geq 0$ uses three sets of disks $\{A_1, A_2, \ldots\}, \{B_1, B_2, \ldots\}$, and $\{C_1, C_2, \ldots\}$, where $i = 1, 2, \ldots$ is the layer index.

For the branch that is symmetric about the x-axis, the construction relies on four horizontal lines [13]:

horizontal line	g	g_1	g_2	g_3
y-value	0	$\sqrt{3}$	$2\sqrt{3}$	$\sqrt{3}+2$

(2)

The disks A_1 and B_1 are aligned in x at heights g_3 and g_1, respectively. All A disks lie on a given convex polygonal chain $\mathcal{A}$ between g_2 and g_3. The chain segments on $\mathcal{A}$ are of length 2 so that subsequent disks A_i and A_{i+1} block each other, and the position of A_1 fixes all other A disks. All C disks lie on g, and C_i blocks B_i from the right (in particular, C_1 is placed after B_1). The disk B_i, for $i > 1$, lies between g and g_1 and it blocks disks A_i and C_{i-1} from the right. With the position of g_2, the branch approaches a hexagonal packing for $i \to \infty$. After reflection about the x-axis, all disks except A_1 and B_1 are locally stable in the infinite branch.

The Böröczky packing is completed by attaching the four branches along the four coordinate axes to a core. For the Böröczky core, both A_1 and B_1 are blocked by core disks (see

Fig. 1a). For the Kahle core, B_1 is blocked by a core disk, and A_1 is locally stable as it also belongs to another branch (see Fig. 1b).

2.1.3 Branches—Finite-Layer Case (Finite *N*), Periodic Boundary Conditions

Branches can also be constructed for periodic simulation boxes, with a finite number k of layers and finite N (see [13]). The branch that connects the central core placed around $(0, 0)$ with its periodic image around $(L, 0)$ is then again symmetric about the x-axis but, in addition, also about the boundary of the central simulation box at $x = L/2$. We describe the construction of the half-branch (for $y \geq 0$) up to this boundary (see Fig. 1).

For half-branches with a finite number of layers k and a finite number of disks $\{A_1, \ldots, A_k\}$, $\{B_1, \ldots, B_k\}$, and $\{C_1, \ldots, C_{k-1}\}$ (with their corresponding mirror images), the convex polygonal chain $\mathcal{A}$ lies between $g_2^<$ and g_3 where $g_2^<$ is an auxiliary horizontal line placed slightly below g_2. The horizontal lines g and g_1 and the algorithm for placing the disks are as in Sect. 2.1.2 (see Fig. 1c, d). By varying the distance between g_2 and $g_2^<$, one can make disk B_k satisfy the additional requirement $x_{B_k} = x_{A_k} + 1$ that allows for periodic boundary conditions. The position of B_k then fixes the boundary of the square box ($x_{B_k} = L/2$) and B_k blocks A_k as well as the mirror image A_{k+1} of A_k (see Fig. 1c again).

2.2 Properties of Böröczky Packings

The local stability of Böröczky packings only relies on the fact that all A disks lie on a largely arbitrary convex polygonal chain $\mathcal{A}$ [13]. The choice of $\mathcal{A}$ influences the qualitative properties

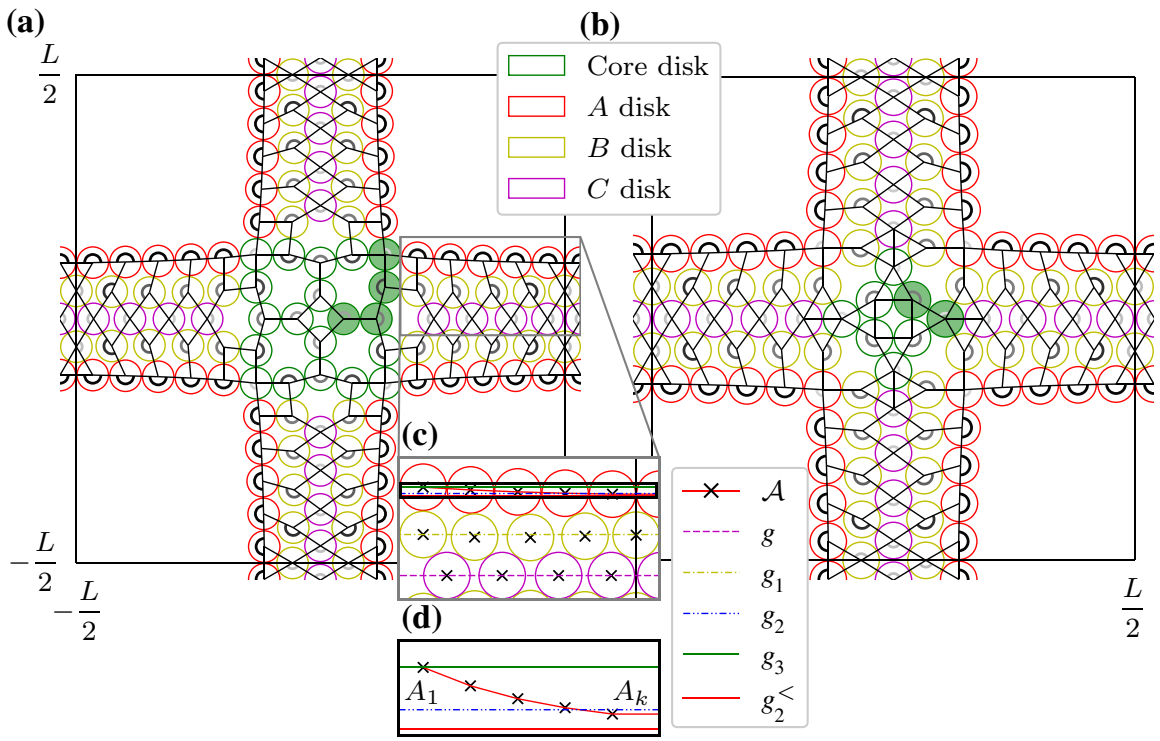

Fig. 1 Hard-disk Böröczky packings, composed of a core and of four branches with $k = 5$ layers, with contact graphs and highlighted opening angles. **a** Packing with the Böröczky core [13]. **b** Packing with the Kahle core [17]. **c** Detail of a branch. **d** Convex polygonal chain $\mathcal{A}$, and horizontal lines $g_2^<$, g_2, and g_3. Two different classes of polygonal chains, called $\mathcal{A}^{\text{geo}}$ and $\mathcal{A}^{\text{circ}}$, are considered in this work

of the packing. The `BigBoro` software package (see Appendix B) implements two different classes of convex polygonal chains that we discuss in Sect. 2.2.1. Another computer program in the package explicitly determines the space of collective escape modes from a Böröczky packing, which we discuss in Sect. 2.2.2.

2.2.1 Convex Polygonal Chains (Geometric, Circular)

In the convex geometric chain $\mathcal{A}^{\text{geo}}$ (which is for instance used in Fig. 1), the disks A_i approach the line $g_2^<$ exponentially in i. In contrast, in the convex circular chain $\mathcal{A}^{\text{circ}}$, all A disks lie on a circle (including their mirror images after reflection about $x = L/2$) so that their opening angles are all the same.

For the convex geometric chain $\mathcal{A}^{\text{geo}}$, the distance between A_i and $g_2^<$ follows a geometric progression:

$$\text{dist}(A_{i+1}, g_2^<) = \phi \ \text{dist}(A_i, g_2^<), \quad \phi \in (0, 1), \tag{3}$$

with the attenuation parameter ϕ. (For a horizontal branch, the distances in Eq. (3) are simply the difference between y-values.) The densities $\eta_{\text{Bör}}$ and η_{Kahle} of the Böröczky packings that either use the Böröczky or the Kahle core vary with ϕ, and they decrease as $\sim 1/k$ for large k (see Table 1). The geometric sequence for A_i induces that the maximum opening angle, usually the one between A_{k-1}, A_k, and A_{k+1}, approaches the angle π as $\theta_k = \phi^{k-2}(1-\phi)(g_3 - g_2^<)/2 \sim \phi^k$, that is, exponentially in k and in L. This implies that the Böröczky packing with the convex geometric chain $\mathcal{A}^{\text{geo}}$ is for large number of layers k exponentially close to losing its local stability (see fifth column of Table 1).

The convex circular chain $\mathcal{A}^{\text{circ}}$ improves the local stability of the Böröczky packing, as the maximum opening angle on $\mathcal{A}$ approaches the critical angle π only algebraically with the number of layers k. Here, all A disks lie on a circle of radius R. This includes A_1, which by construction lies on g_3 (see Sect. 2.1.2). The circle is tangent to $g_2^<$ at $x = L/2$. The center

Table 1 Parameters of Böröczky packings for different numbers k of layers with $N \sim 20k$ given by Eq. (4)

Layers k	Density $\eta_{\text{Bör}}$	Density η_{Kahle}	Def. angle$^{\text{circ}}$	Def. angle$^{\text{geo}}$
5	$0.3957 \pm 3.1\times10^{-4}$	$0.4660 \pm 4.3\times10^{-4}$	8.3×10^{-1}	3.8×10^{-1}
6	$0.3625 \pm 2.9\times10^{-4}$	$0.4204 \pm 3.9\times10^{-4}$	5.3×10^{-1}	2.5×10^{-1}
7	$0.3338 \pm 2.6\times10^{-4}$	$0.3820 \pm 3.3\times10^{-4}$	3.8×10^{-1}	1.8×10^{-1}
8	$0.3089 \pm 2.2\times10^{-4}$	$0.3496 \pm 2.8\times10^{-4}$	2.8×10^{-1}	1.3×10^{-1}
9	$0.2873 \pm 1.9\times10^{-4}$	$0.3219 \pm 2.4\times10^{-4}$	2.2×10^{-1}	9.9×10^{-2}
10	$0.2683 \pm 1.7\times10^{-4}$	$0.2982 \pm 2.1\times10^{-4}$	1.7×10^{-1}	7.6×10^{-2}
15	$0.2010 \pm 9.5\times10^{-5}$	$0.2171 \pm 1.1\times10^{-4}$	7.3×10^{-2}	2.2×10^{-2}
20	$0.1604 \pm 6.0\times10^{-5}$	$0.1704 \pm 6.7\times10^{-5}$	4.1×10^{-2}	7.0×10^{-3}
30	$0.1141 \pm 3.0\times10^{-5}$	$0.1190 \pm 3.2\times10^{-5}$	1.8×10^{-2}	7.4×10^{-4}
50	$0.0722 \pm 1.2\times10^{-5}$	$0.0741 \pm 1.2\times10^{-5}$	6.3×10^{-3}	8.5×10^{-6}
100	$0.0376 \pm 3.1\times10^{-6}$	$0.0381 \pm 3.2\times10^{-6}$	1.6×10^{-3}	1.2×10^{-10}
1000	$0.0039 \pm 3.3\times10^{-8}$	$0.0039 \pm 3.3\times10^{-8}$	1.5×10^{-5}	7.4×10^{-98}

Second and third columns: Density window for the Böröczky and Kahle cores with $\mathcal{A}^{\text{geo}}$, obtained from ϕ between 0.0001 and 0.9. Fourth and fifth columns: Deficit angle with respect to 180° of the maximum opening angle (in degrees, same for both cores) for $\mathcal{A}^{\text{circ}}$ and for $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.8$

of the circle lies on the vertical line at $x = L/2$. It follows from elementary trigonometry that for large k, the radius of the circle R scales as $\sim k^2$ and that the maximum opening angle approaches the angle π as $\sim k^{-2}$ (see fourth column of Table 1).

2.2.2 Contact Graphs: Local and Collective Stability

The contact graph of a Böröczky packing connects any two disks whose pair distance equals 2 (including periodic boundary conditions, see Fig. 1). In a Böröczky packing with $k \geq 1$ layers, the number N of disks and the number N_{contact} of contacts are:

	N	N_{contact}
Bpöröczky core	$20k + 12$	$32k + 20$.
Kahle core	$20k - 4$	$32k + 4$

(4)

For all values of $k > 1$, the number of contacts is smaller than $2N - 2$. This implies that collective infinitesimal two-dimensional displacements, with $2N - 2$ degrees of freedom (the values of the displacements in x and in y for each disk avoiding trivial translations), can escape from a Böröczky packing, which is thus not collectively stable [17].

When all disks i, at positions $\mathbf{x}_i = (x_i, y_i)$, are moved to $\mathbf{x}_i + \boldsymbol{\Delta}_i$ with $\boldsymbol{\Delta}_i = (\Delta_i^x, \Delta_i^y)$, the squared separation between two touching disks from the contact graph i and j changes from $|\mathbf{x}_i - \mathbf{x}_j|^2$ to

$$|\mathbf{x}_i + \boldsymbol{\Delta}_i - (\mathbf{x}_j + \boldsymbol{\Delta}_j)|^2 = |\mathbf{x}_i - \mathbf{x}_j|^2 + \underbrace{2(\mathbf{x}_i - \mathbf{x}_j) \cdot (\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j)}_{\text{first-order variation}} + |\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j|^2. \tag{5}$$

If the first-order term in Eq. (5) vanishes for all contacts i and j, the separation between touching disks cannot decrease. It then increases to second order in the displacements, if $\boldsymbol{\Delta}_i \neq \boldsymbol{\Delta}_j$, so that contact is lost. Distances between disks that are not in contact need not be considered because the displacements $\boldsymbol{\Delta}_i$ are infinitesimal. The first-order variation in Eq. (5) can be written as a product of twice an "escape matrix" $\mathcal{M}^{\text{esc}}$ of dimensions $N_{\text{contacts}} \times 2N$ with a $2N$-dimensional vector $\boldsymbol{\Delta} = (\Delta_1^x, \Delta_1^y, \Delta_2^x, \Delta_2^y, \ldots)$. The row r of $\mathcal{M}^{\text{esc}}$ corresponding to the contact between i and j has four non-zero entries

$$\begin{aligned} \mathcal{M}^{\text{esc}}_{r,2i-1} &= x_i - x_j, \\ \mathcal{M}^{\text{esc}}_{r,2i} &= y_i - y_j, \\ \mathcal{M}^{\text{esc}}_{r,2j-1} &= -(x_i - x_j), \\ \mathcal{M}^{\text{esc}}_{r,2j} &= -(y_i - y_j). \end{aligned} \tag{6}$$

The `BigBoro` software package (see Appendix B) solves for

$$\mathcal{M}^{\text{esc}} \boldsymbol{\Delta} = 0 \tag{7}$$

using singular-value decomposition. The solutions of Eq. (7) are the directions of the small displacements that break the contacts but do not introduce overlapping disks. For the $k = 5$ Böröczky packing with the Kahle core, we find 28 vanishing singular values. It follows from Eq. (4) that, because of $28 = 2N - N_{\text{contact}}$, all contacts are linearly independent. We classify the 28 modes by studying the following cost function on the contact graph:

$$L = \sum_{i,j} (\boldsymbol{\Delta}_i - \boldsymbol{\Delta}_j)^2, \tag{8}$$

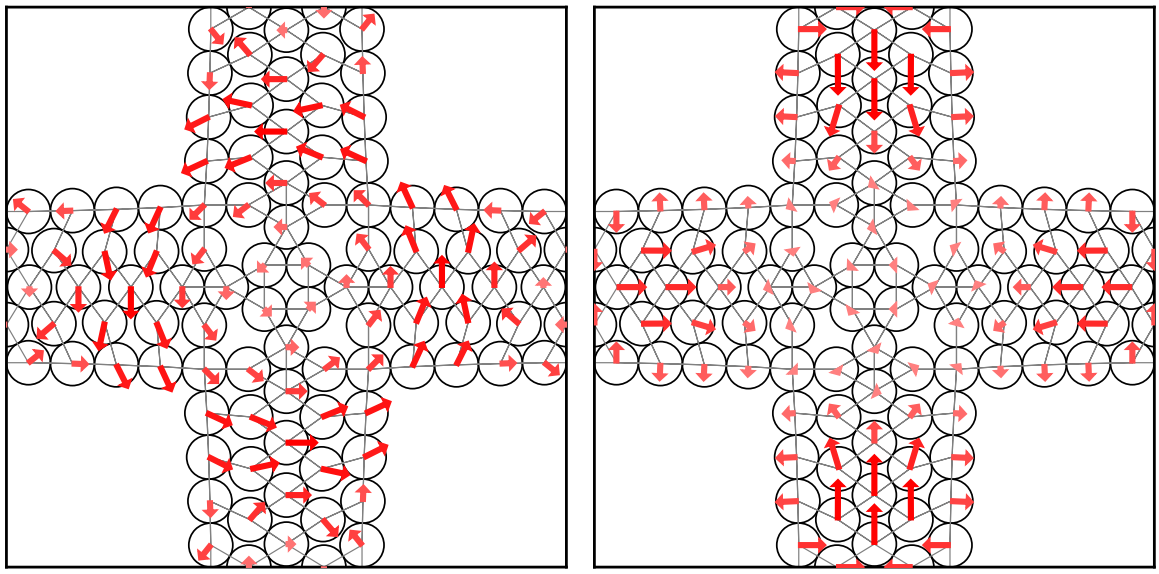

Fig. 2 Two orthogonal modes (represented as red arrows) out of the 28-dimensional space of all collective escape modes $\boldsymbol{\Delta}$ for the $k = 5$ Böröczky packing with the Kahle core and the convex geometric chain $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.7$. Lines are drawn between pairs of disks which are in contact

where the sum is over all contact pairs i and j. This function, acting on the $2N$ displacements $\boldsymbol{\Delta}$, measures the non-uniformity of a deformation. It acts as a quadratic form within the 28-dimensional space of vanishing singular values, and can be diagonalized within this space. The resulting two lowest eigenmodes (with zero eigenvalue) of Eq. (8) describe rigid translation of the packing in the plane. Other low-lying eigenmodes give smooth large-scale deformations which collectively escape the contact constraints (see Fig. 2).

For $k \geq 1$, the number of contacts in Eq. (4) is larger than $N - 1$. Böröczky packings are thus collectively stable for displacements that are constrained to a single direction, as for example the x or y direction. This strongly constrains the dynamics of MCMC algorithms that for a certain time have only one degree of freedom per disk.

2.2.3 Dimension of the Space of Böröczky Packings

As discussed in Sect. 2.2.2, each Böröczky packing has a contact graph. Conversely, a given contact graph describes Böröczky packings for a continuous range of densities η. As an example, changing the attenuation parameter ϕ of the convex polygonal chain $\mathcal{A}^{\text{geo}}$ in Eq. (3) continuously moves all branch disks, and in particular disk B_k and, therefore, the value of L and the density η (see Table 1 for density windows that can be obtained in this way). We conjecture that locally stable packings exist for any density at large enough N. Sparse locally stable packings can also be part of dense hard-disk configurations where the majority of disks are free to move.

Moreover, the space $\mathcal{B}$ of locally stable packings of N disks of radius σ in a given central simulation box is of lower dimension than the sample space Ω: For each contact graph, each independent edge decreases the dimensionality by one. In addition there is only a finite number of contact graphs for a given N. The low dimension of $\mathcal{B}$ also checks with the fact that any packing, and more generally, any configuration with contacts, has effectively infinite pressure (see the detailed discussion in Sect. 4.2.2). As the ensemble-averaged pressure is finite (except for the densest packing), the packings (and the configurations containing packings) must be of lower dimension. As the dimension of $\mathcal{B}$, for large N, is much lower than that of Ω, we conjecture $\Omega \setminus \mathcal{B}$ to be connected for a given η below the densest packing at large enough N although, in our understanding, this is proven only for $\eta \sim 1/\sqrt{N}$ (see [19, 20]).

3 MCMC Algorithms and ε-relaxed Böröczky configurations

In this section, we first introduce to a number of local MCMC algorithms (see Sect. 3.1). In Sect. 3.2, we then determine the escape times (in the number of trials or events) after which these algorithms escape from ε-relaxed Böröczky configurations, that is, from Böröczky packings with disk radii multiplied by a factor $(1 - \varepsilon)$ (see Fig. 3a, b). A scaling theory establishes the existence of two classes of MCMC algorithms, one in which the escape time from an ε-relaxed Böröczky configurations scales algebraically with ε, with exponents that are predicted exactly, and the other in which the scaling is logarithmic. Numerical simulations confirm the theory.

3.1 Local Hard-Disk MCMC Algorithms

We define the reversible Metropolis algorithm with two displacement sets, from which the trial moves are uniformly sampled (see Sect. 3.1.1). We also consider variants of the non-reversible ECMC algorithm that only differ in their treatment of events, that is, of disk collisions (see Sect. 3.1.2). An arbitrary-precision implementation of the discussed ECMC algorithms (in the Go programming language) is contained in the `BigBoro` software package (see Appendix B).

3.1.1 Local Metropolis Algorithm: Displacement Sets

The N disks are at positions $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$. In the local Metropolis algorithm [1], at each time $t = 1, 2, \ldots$, a trial move is proposed for a randomly chosen disk i, from its position $\mathbf{x}_i$ to $\mathbf{x}_i + \Delta\mathbf{x}_i$. If the trial produces an overlap, disk i stays put and $\mathbf{x}$ remains unchanged. We study two sets for the trial moves. For the cross-shaped displacement set, the trial moves are uniformly sampled within a range δ along the coordinate axes, that is, either along the x-axis ($\Delta\mathbf{x}_i = (\mathtt{ran}(-\delta, \delta), 0)$) or along the y-axis ($\Delta\mathbf{x}_i = (0, \mathtt{ran}(-\delta, \delta))$). Alternatively, for the square-shaped displacement set, the trial moves are uniformly sampled as $\Delta\mathbf{x}_i = (\mathtt{ran}(-\delta, \delta), \mathtt{ran}(-\delta, \delta))$. A Böröczky packing traps the local Metropolis algorithm if the range δ is smaller than a critical range δ_c. This range is closely related to the maximum opening angle (see the discussion in Sect. 2.2.1 and Fig. 3c). For these packings, the critical range vanishes for $N \to \infty$ independently of the specific core or of the convex polygonal chain, simply because the maximum opening angle approaches π in that limit. On the other hand, for large range δ, the algorithm can readily escape from the stable configuration. For $\delta = L/2$, the Metropolis algorithm with a square-shaped displacement set proposes a random placement of the disk i inside the central simulation box. This displacement set leads to a very inefficient algorithm at the densities of physical interest, but it mixes very quickly at small finite densities (see Sect. 4.2.1). For the scaling theory of the escape of the Metropolis algorithm from ε-relaxed Böröczky configurations, we consider ranges δ smaller than the critical range δ_c.

3.1.2 Hard-Disk ECMC: Straight, Reflective, Forward, Newtonian

Straight ECMC [5] is one of the two original variants of event-chain Monte Carlo. This Markov chain evolves in (real-valued) continuous Monte-Carlo time t_{MCMC}, but its implementation is event-driven. The algorithm is organized in a sequence of "chains", each with a chain time τ_{chain}, its intrinsic parameter. In each chain, with Monte-Carlo time between

t_{MCMC} and $t_{\mathrm{MCMC}} + \tau_{\mathrm{chain}}$, disks move with unit velocity in one given direction (alternatively in $+x$ or in $+y$). A randomly sampled initial disk thus moves either until the chain time τ_{chain} is used up, or until, at a collision event, it collides with another disk, which then moves in its turn, etc. This algorithm is highly efficient in some applications [4, 5, 21]. During each chain (in between changes of direction), any disk can collide only with three other disks or fewer [22, 23]. A constraint graph with directed edges may encode these relations. This constraint graph (defined for hard-disk configurations) takes on the role of the contact graph (that is defined for packings) (see Fig. 3a, b). As the moves in a chain are all in the same direction, straight ECMC has only $N - 1$ degrees of freedom, fewer than there are edges in the constraint graph. It is for this reason that it may encounter the rigidity problems evoked in Sect. 2.2.2.

In reflective ECMC [5], in between events, disks move in straight lines with unit velocity just as in straight ECMC. At a collision event, the target disk does not continue in the same direction as the active disk. Rather, the target-disk direction is the original active-disk direction reflected from the line connecting the two disk centers at contact (see [5]). As all ECMC variants, reflective ECMC satisfies the global-balance condition. Irreducibility (for connected sample spaces) requires in principle resamplings of the active disk and its velocity in intervals of the chain time τ_{chain} [15, 24, 25]. However, this seems not always necessary [15, 25]. Numerical experiments indicate that reflective ECMC requires no resamplings in our case as well. It is also faster without them (see Appendix A.2). A variant of reflective ECMC, obtuse ECMC [16], has shown interesting behavior.

Forward ECMC [15] updates the normalized target-disk direction as follows after an event. The component orthogonal to the line connecting the disks at contact is uniformly sampled between 0 and 1 (reflecting the orthogonal orientation). Its parallel component is determined so that the direction vector (which is also the velocity vector) is of unit norm. The parallel orientation remains unchanged. In contrast to reflective ECMC, the event-based randomness renders forward ECMC practically irreducible for the considered two-dimensional hard-disk systems even without resamplings. Resamplings in intervals of the chain time τ_{chain} can still be considered but slow the algorithm down (see Appendix A.2). We thus consider forward ECMC without resampling.

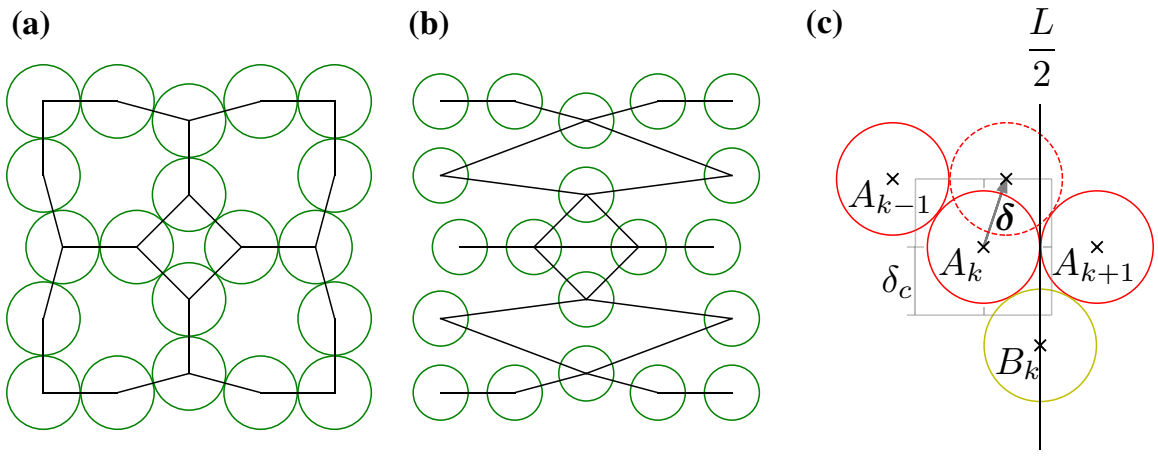

Fig. 3 Contact graphs, constraint graphs and minimal escape range. **a** Contact graph for a packing consisting solely of the Böröczky core. **b** Constraint graph in x-direction for an ε-relaxed Böröczky configurations derived from the same packing with $\varepsilon = 0.25$. The edges indicate all possible collisions of straight ECMC in x-direction. **c** Escape move $\boldsymbol{\delta}$ and minimal escape range δ_c of the Metropolis algorithm with a square-shaped displacement set

Newtonian ECMC [16] mimics molecular dynamics in order to determine the velocity of the target disk in an event. It initially samples disk velocities from the two-dimensional Maxwell distribution with unit root-mean-square velocity. However, at each moment, only a single disk is actually moving with its constant velocity. At a collision event, the velocities of the colliding disks are updated according to Newton's law of elastic collisions for hard disks of equal masses, but only the target disks actually moves after the event. In this algorithm, the velocity (which indexes the Monte-Carlo time) generally differs from unity. Similar to reflective ECMC, we tested that resamplings appear not to be required in our case (and again yield a slower performance, see Appendix A.2), although Newtonian ECMC manifestly violates irreducibility in highly symmetric models [25]. As in earlier studies for three-dimensional hard-sphere systems [16] and for two-dimensional dipoles [25], Newtonian ECMC is typically very fast for ε-relaxed Böröczky configurations. However, it suffers from frequent gridlocks (see Sect. 3.2.4).

3.2 Escape Times from ε-relaxed Böröczky configurations

The principal figure of merit for a Markov chain is its mixing time [26], the number of steps it takes from the worst-case initial condition to approach the stationary probability distribution to some precision level. Böröczky packings trap the local Metropolis dynamics (of sufficiently small range) as well as ECMC dynamics, so that the mixing time is, strictly speaking, infinite. Although they cannot be escaped from, the packings make up only a set of measure zero in sample space, and might thus be judged irrelevant.

However, as we will discuss in the present subsection, the situation is more complex. For every Böröczky packing, an associated ε-relaxed Böröczky configurations keeps the central simulation box and the disk positions, but reduces the disk radii from 1 to $1-\varepsilon$. An ε-relaxed Böröczky configurations effectively defines a finite portion of the sample space (the spheres of radius ε around each disk position of the packing). All MCMC algorithms considered in this work escape from these configurations in an escape time that diverges as $\varepsilon \to 0$ (see Sect. 3.2.1 for a definition of escape times). Numerical results and a scaling theory for the escape times are discussed in Sects. 3.2.2 and 3.2.3, and a synopsis of our results is contained in Sect. 3.2.4. The divergent escape times as $\varepsilon \to 0$ are specific to the NVT ensemble (as we will discuss in Sect. 4.2.2).

3.2.1 Nearest-Neighbor Distances and Escape Times

In a Böröczky packing, disks are locally stable, and they all have a nearest-neighbor distance of 2. The packings are sparse, and the nearest-neighbor distance is thus smaller than its $\sim 1/\sqrt{\eta}$ equilibrium value. To track the escape from an ε-relaxed Böröczky configurations, we monitor the maximum nearest-neighbor distance:

$$d(t) = \max_i \left[\min_{j(\neq i)} |\mathbf{x}_{ij}(t)| \right], \tag{9}$$

where $|\mathbf{x}_{ij}(t)| = |\mathbf{x}_j(t) - \mathbf{x}_i(t)|$ is the distance between disks i and j (possibly corrected for periodic boundary conditions). The maximum nearest-neighbor distance signals when a single disk breaks loose from what corresponds to its contacts. In the further time evolution, the configuration then falls apart. For the Metropolis algorithm, we compute $d(t)$ once every N trials, and t denotes the integer-valued number of individual trial moves. For ECMC, we sample $d(t)$ and the number of events in intervals of the sampling Monte-Carlo time. In

Eq. (9), t then denotes the integer-valued number of events. Both discrete times t increment by one with a computational effort $\mathcal{O}(1)$, corresponding to one trial in the Metropolis algorithm and to one event in ECMC. Starting from an ε-relaxed Böröczky configurations, $d(t)$ typically remains at $d(t) \sim 2 + \mathcal{O}(\varepsilon)$ for a long time until it approaches the equilibrium value in a way that depends on the algorithm. We define the escape time t_{esc}, an integer, as the time t at which $d(t)$ has increased by ten percent:

$$t_{\text{esc}} = \min\left[t : d(t) > 2(1+\gamma)\right], \tag{10}$$

with $\gamma = 0.1$. All our results for the scaling of the escape time with the relaxation parameter ε in the following subsections were reproduced for $\gamma = 0.025$ (see Appendix A.1). The definition of the escape time based on the maximum nearest-neighbor distance $d(t)$ is certainly not the only one to monitor the stability of ε-relaxed Böröczky configurations. It may not be equally well-suited for all considered algorithms. Still, our scaling theory suggests that the algorithms with an intrinsic parameter show a distinctly different behavior than the algorithms without them, which appears to be independent of the precise definition of the escape time.

3.2.2 Escape-Time Scaling for Metropolis and Straight ECMC

The local Metropolis algorithm and straight ECMC both have an intrinsic parameter, namely the range δ of the displacement set or the chain time τ_{chain}. These two parameters play a similar role. We numerically measure the escape time t_{esc} of these algorithms for a wide range of their intrinsic parameters and for small relaxation parameters ε (see Fig. 4, for the escape times from ε-relaxed Böröczky configurations with $k = 5$ layers and the Kahle core). The escape time diverges for $\delta, \tau_{\text{chain}} \to 0$. For straight ECMC and small ε, t_{esc} also diverges for $\tau_{\text{chain}} \to \infty$ so that the function is "V"-shaped with an optimal chain time $\tau_{\text{chain}}^{\min}$. For the Metropolis algorithm, t_{esc} increases until around the critical range δ_c so that there is an optimal range $\delta^{\min} < \delta_c$.

Two limiting cases can be analyzed in terms of the intrinsic parameter $\delta < \delta_c$ or τ_{chain}, and the internal length scales ε, and σ. For the Metropolis algorithm at small δ, a trajectory

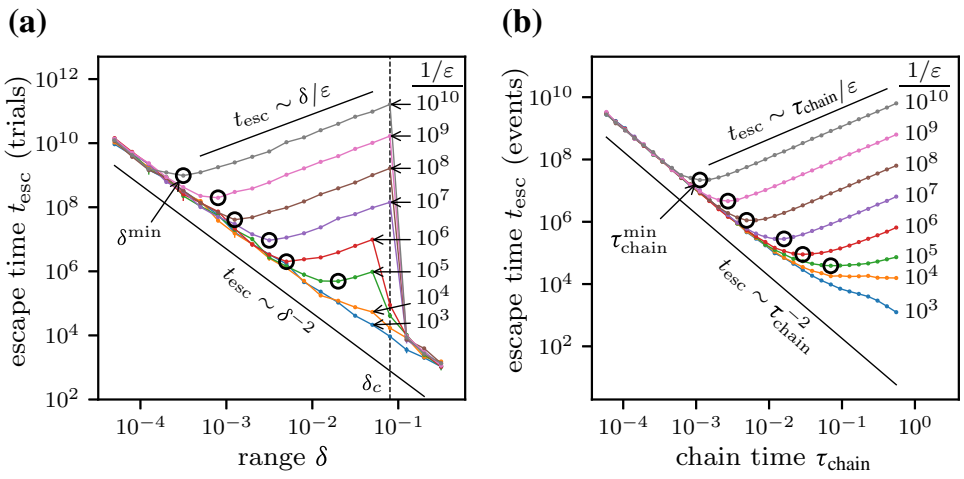

Fig. 4 Median escape times from the $k = 5$ ε-relaxed Böröczky configurations (Kahle core and convex geometric chain $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.7$, $N = 96$ disks) for different ε. **a** t_{esc} (in trials) *vs.* range δ for the Metropolis algorithm with the cross-shaped displacement set. **b** t_{esc} (in events) *vs.* chain time τ_{chain} for straight ECMC. Asymptotes are from Eqs. (11) and (12). Error bars are smaller than the marker sizes

spanning a constant distance is required to escape from an ε-relaxed Böröczky configurations. This constant distance can be thought of as the escape distance δ_c in Fig. 3, which is on a scale σ and independent of ε for small ε. As the Monte-Carlo dynamics is diffusive, this constant distance satisfies const $= \delta\sqrt{t_{\text{esc}}}$. For straight ECMC with small chain times τ_{chain}, the effective dynamics (after subtraction of the uniform displacement), is again diffusive. This leads to:

$$t_{\text{esc}} \sim \begin{cases} \text{const}/\delta^2 & \text{(Metropolis)}, \\ \text{const}/\tau_{\text{chain}}^2 & \text{(straight ECMC)}, \end{cases} \quad \text{(for small } \delta < \delta_c, \tau_{\text{chain}}). \tag{11}$$

The independence of t_{esc} of the relaxation parameter ε for small intrinsic parameters is clearly brought out in the numerical simulations (see Fig. 4).

On the other hand, even for large $\delta < \delta_c$ or τ_{chain}, the Markov chain must make a certain number of moves on a length scale ε in order to escape from the ε-relaxed Böröczky configurations. In the Metropolis algorithm, the probability for a trial on this scale is ε/δ for the cross-shaped displacement set and ε^2/δ^2 for the square-shaped displacement set. For the straight ECMC with large τ_{chain}, all displacements beyond a time $\sim \varepsilon$ (or, possibly, $\sim N\varepsilon$) effectively cancel each other, because the constraint graph is rigid. This leads to:

$$t_{\text{esc}} \sim \begin{cases} \delta^2/\varepsilon^2 & \text{(Metropolis—square)}, \\ \delta/\varepsilon & \text{(Metropolis—cross)}, \\ \tau_{\text{chain}}/\varepsilon & \text{(straight ECMC)}, \end{cases} \quad \text{(for large } \delta < \delta_c, \tau_{\text{chain}}). \tag{12}$$

The scaling of t_{esc} as $\sim 1/\varepsilon$ or $\sim 1/\varepsilon^2$ for large intrinsic parameters is confirmed in the numerical simulations for small relaxation parameters ε (see Fig. 4). For large ε, the critical range δ_c of the Metropolis algorithm (that slightly decreases with ε) falls below the region of large δ. For large ε, the constraint graph of straight ECMC loses its rigidity, and τ_{chain} no longer appears as a relevant intrinsic parameter. The scaling theory no longer applies.

The two asymptotes of Eqs. (11) and (12) form a "V" with a base $\delta^{\min}$ (or $\tau_{\text{chain}}^{\min}$) that is obtained by equating the two expressions for $t_{\text{esc}}(\delta)$ (or $t_{\text{esc}}(\tau_{\text{chain}})$). This yields $\delta^{\min} \sim \sqrt[3]{\varepsilon}$ for the Metropolis algorithm with a cross-shaped displacement set, and likewise $\tau_{\text{chain}}^{\min} \sim \sqrt[3]{\varepsilon}$ for straight ECMC. For the Metropolis algorithm with a square-shaped move set, one obtains $\delta^{\min} \sim \sqrt{\varepsilon}$. The resulting optimum, the minimal escape time with respect to ε, is

$$t_{\text{esc}} \sim \begin{cases} \varepsilon^{-1} & \text{(Metropolis—square)}, \\ \varepsilon^{-2/3} & \text{(Metropolis—cross)}, \\ \varepsilon^{-2/3} & \text{(straight ECMC)}, \end{cases} \quad \text{(for optimal } \delta^{\min}, \tau_{\text{chain}}^{\min}). \tag{13}$$

These scalings balance two requirements: to move by a constant distance (which favors large δ or τ_{chain}) and to move on the scale ε (which favors small δ or τ_{chain}).

3.2.3 Time Dependence of Free Path—Reflective, Forward, and Newtonian ECMC

The forward, reflective, and Newtonian variants of ECMC move in any direction, even in the absence of resamplings, so that their displacement sets are $2N$-dimensional. This avoids the rigidity problem of straight ECMC (the fact that the number of constraints can be larger than the number of degrees of freedom). We consider these algorithms without resamplings, that is, for $\tau_{\text{chain}} = \infty$. Finite chain times yield larger escape times that approach the value at $\tau_{\text{chain}} = \infty$ (see Appendix A.2). Without an intrinsic parameter, the effective free path between events may thus adapt as the configuration gradually escapes from the ε-relaxed

Böröczky configurations. The free path is initially on the scale ε, but then grows on average by a constant factor at each event, reaching a scale $\varepsilon' > \varepsilon$ after a time (that is, after a number of events) that scales as $\sim \ln(\varepsilon'/\varepsilon)$. The scale ε' at which the algorithms break free is independent of the initial scale ε, and we expect a logarithmic scaling of the escape time (measured in events):

$$t_{\mathrm{esc}} \sim \ln(1/\varepsilon) \quad \text{(reflective, forward, and Newtonian ECMC).} \tag{14}$$

The absence of an imposed scale for displacements manifests itself in the logarithmic growth with time of the average free path, that is, the averaged displacement between events over many simulations starting from the same ε-relaxed Böröczky configurations (see Fig. 5 for the example of the escape of forward ECMC from ε-relaxed Böröczky configurations with $k = 5$ layers and the Kahle core). Individual evolutions as a function of time t for small relaxation parameters ε and ε' overlap when shifted by their escape times. Starting from an ε-relaxed Böröczky configurations with $\varepsilon = 10^{-30}$, as an example, the same time is on average required to move from an average free path of $\sim 10^{-30}$ to 10^{-25}, as from an average free path $\sim 10^{-25}$ to 10^{-20}. The time t in this discussion refers to the number of events and not to the Monte-Carlo time t_{MCMC}. As discussed, the velocity in reflective and forward ECMC, and the root-mean-square velocity in Newtonian ECMC, have unit value. The free path between subsequent events—which, as discussed, grows exponentially with t—then equals the difference of Monte-Carlo times $t_{\mathrm{MCMC}}(t+1) - t_{\mathrm{MCMC}}(t)$. The Monte-Carlo time t_{MCMC} thus grows as a geometric series and depends exponentially on the number of events t. This emphasizes that the escape from an ε-relaxed Böröczky configurations is a non-equilibrium phenomenon.

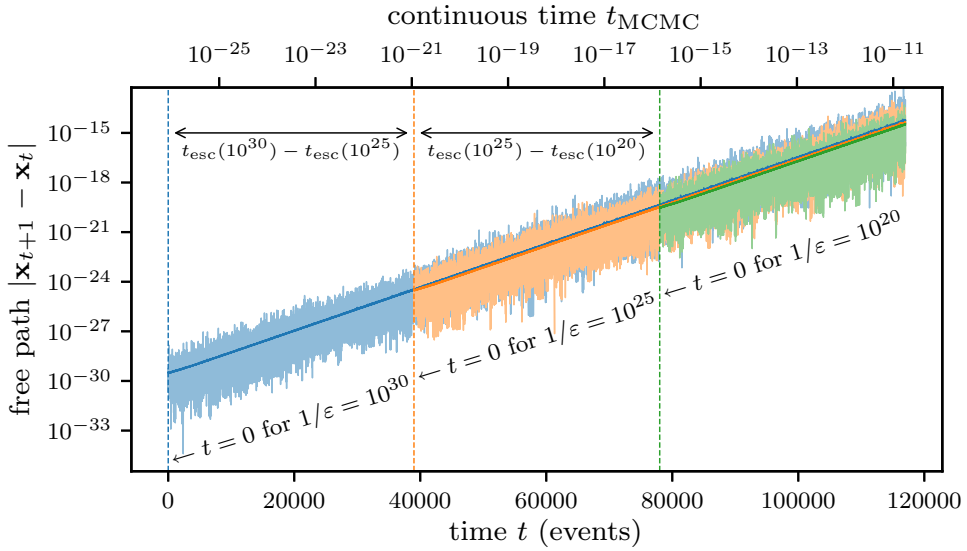

Fig. 5 Free path (equivalently: Monte-Carlo time between events) for the forward ECMC algorithm started from three $k = 5$ ε-relaxed Böröczky configurations (Kahle core and convex geometric chain $\mathcal{A}^{\mathrm{geo}}$ with attenuation parameter $\phi = 0.7$, $N = 96$ disks) with $\varepsilon = 10^{-30}$, 10^{-25} and 10^{-20}. Integer time t (lower x-axis) counts events, while t_{MCMC} (upper x-axis) is the real-valued continuous Monte-Carlo time. Event times are shifted. Expanded light curves show single simulations for each ε, dark lines average over 10,000 simulations

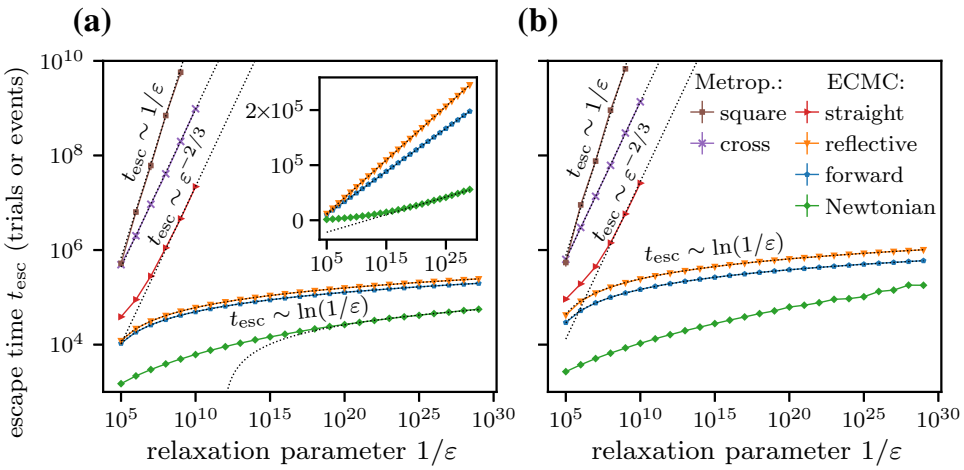

Fig. 6 Median escape time t_{esc} from $k = 5$ ε-relaxed Böröczky configurations with different cores (with convex geometric chain $\mathcal{A}^{\mathrm{geo}}$ and attenuation parameter $\phi = 0.7$) for local MCMC algorithms (where applicable: with optimized intrinsic parameters). **a** a t_{esc} for the Kahle core ($N = 96$ disks). The Metropolis algorithm and straight ECMC show an algebraic scaling. Inset: log–lin plots suggesting logarithmic scaling for the forward, reflective, and Newtonian ECMC. **b** t_{esc} for the Böröczky core ($N = 112$ disks). Newtonian ECMC has frequent gridlocks for small ε so that its logarithmic scaling is distorted. Error bars are smaller than the marker sizes.

3.2.4 Escape Times: Synopsis of Numerical Results and Scaling Theory

Overall, escape times $t_{\mathrm{esc}}(\varepsilon)$ (with intrinsic parameters optimized through a systematic scan for the Metropolis algorithm and for straight ECMC) validate the algebraic scalings of Eq. (13), on the one hand, and the logarithmic scaling of Eq. (14), on the other (see Fig. 6 for the escape times from ε-relaxed Böröczky configurations with $k = 5$ layers with either the Kahle core or the Böröczky core). Our arbitrary-precision implementation of reflective, forward, and Newtonian ECMC confirms their logarithmic scaling down to $\varepsilon = 10^{-29}$. Newtonian ECMC appears *a priori* as the fastest variant of ECMC. However, it frequently gets gridlocked, i.e., trapped in circles of repeatedly active disks with a diverging event rate. Gridlocks also rarely appear in straight and reflective ECMC. In runs that end up in gridlock, escape times are very large, possibly diverging. (In Figs. 4 and 6, median escape times rather than the means are therefore displayed for all algorithms. Mean and median escape times are similar for the Metropolis algorithm and forward ECMC where gridlocks play no role.) The gridlock rate increases with $1/\varepsilon$. For the Kahle core, this effect is negligible for all ε. For the Böröczky core, the gridlock rate of Newtonian ECMC is $\sim 30\%$ for $\varepsilon = 10^{-29}$ (see Fig. 6b, the logarithmic scaling is distorted even for the median). We observe no clear dependence of the gridlock rate on the floating-point precision of our arbitrary-precision ECMC implementation, and it thus appears unlikely that gridlocks are merely numerical artifacts (see Appendix A.3).

Gridlock is the very essence of ECMC dynamics from a locally stable Böröczky packing, but it can also appear as a final state from an ε-relaxed Böröczky configurations. We observe gridlocks in all hard-disk ECMC variants that feature deterministic collision rules. They were previously observed for straight ECMC from tightly packed initial configurations [27, Sect. 4.2.3]. Only forward ECMC with its event-based randomness is free of them. In a gridlock, the event rate diverges at a given Monte-Carlo time, which then seems to stand still so that no finite amount of Monte-Carlo time is spent in a configuration with contacts. Because of the divergence of the event rate, gridlocks cannot be cured through resamplings at fixed

Monte-Carlo-time intervals. To overcome them in Newtonian ECMC, which appears *a priori* as the fastest of our ECMC variants, one can probably introduce event-based randomness as is done in forward ECMC. Nevertheless, gridlocks play no role in large systems at reasonable densities. Also, ECMC algorithms for soft potentials introduce randomness at each event so that gridlocks should not appear.

4 Discussion

In the present section, we discuss our results for the escape times (Sect. 4.1), as well as a number of more fundamental aspects of Böröczky packings in the context of MCMC (Sect. 4.2). We in particular clarify why a packing effectively realizes an infinite-pressure configuration that in a constant-pressure Monte-Carlo simulation is instantly relaxed through a volume increase.

4.1 Escape Times: Speedups, Bottlenecks

ECMC is a continuous-time MCMC method, and its continuous Monte-Carlo time t_{MCMC} takes the place of the usual count of discrete-time Monte-Carlo trials. However, ECMC is event-driven. The time t, and especially the escape time t_{esc}, are integers, and they count events. The computational effort in hard-disk ECMC is $\mathcal{O}(1)$ per event, using a cell-occupancy system that is also implemented in the `BigBoro` software package. In several of our algorithms, the times t and t_{MCMC} are not proportional to each other, because the free path (roughly equivalent to the Monte-Carlo time between events) evolves during each individual run.

4.1.1 Range of Speedups

The speedup realized by lifted Markov chains, of which ECMC is a representative, corresponds to the transition from diffusive to ballistic transport [6, 28, 29]. This speedup refers to what we call the "Monte-Carlo time" t_{MCMC}, that is the underlying time of the Markov process, and not to the time t that is measured in events. For Markov chains in a finite sample space Ω, the Monte-Carlo time for mixing of the lifted Markov chain cannot be smaller than the square root of the mixing time for the original (collapsed) chain. The remarkable power-law-to-logarithm speedup in ε realized by some of the ECMC algorithms concerns escape times which measure the number of events. The Monte-Carlo escape times probably conform to the mathematical bounds, although it is unclear how to approximate hard-disk MCMC for $\varepsilon \to 0$ through a finite Markov chain. Mathematical results for the Monte-Carlo escape times from locally blocked configurations would be extremely interesting, even for models with a restricted number of disks.

4.1.2 Space of ε-Relaxed Böröczky Configurations

The definition of an ε-relaxed Böröczky configurations can be generalized. Equivalent legal hard-disk configurations are obtained by reducing the disk radii and choosing random disk positions in a circle of radius ε around the original disk positions in the Böröczky packing. These configurations also feature the escape-time scalings given in Eqs. (13) and (14). Any ε-relaxed Böröczky configurations is thus merely a sample in a space $\mathcal{B}_\varepsilon$ of volume $\sim \varepsilon^{2N}$.

For a given upper limit t_{cpu} of CPU time at fixed N, this corresponds to a volume of $\mathcal{B}_\varepsilon$ (that cannot be escaped from in t_{cpu}) scaling with the computer-time budget as $\sim t_{\mathrm{cpu}}^{-3N}$ for the straight ECMC and scaling as $\sim \exp\left(-2Nt_{\mathrm{cpu}}\right)$ for the forward ECMC. We expect $\mathcal{B}_\varepsilon$ to have a double role, as a space of configurations that the Monte-Carlo dynamics cannot practically escape from, but maybe also a space that it cannot even access. The volume of $\mathcal{B}_\varepsilon$ (with ε chosen such that it cannot be escaped from in a reasonable CPU time) as well as the corresponding changes in the free energy per disk are probably unmeasurably small except, possibly, at very small N. The existence of a finite fraction of sample space that cannot be escaped from in any reasonable CPU time at finite N is however remarkable. In many MCMC algorithms for physical systems, as for example the Ising model, parts of sample space are practically excluded because of their low Boltzmann weight, but they feature diverging escape times only in the limit $N \to \infty$.

In this context, we note that Markov chains can be interpreted in terms of a single bottleneck partitioning the sample space into two pieces [26, Sect. 7.2]. The algorithmic stationary probability flow across the bottleneck sets the conductance of an algorithm, which again bounds mixing and correlation times. Ideally, MCMC algorithms would be benchmarked through their conductances. In the hard-disk model, the bottleneck has not been identified, so that the benchmarking and the analysis of MCMC algorithms must rely on empirical criteria. However, Böröczky packings and the related ε-relaxed Böröczky configurations may well model a bottleneck, from which the Markov chain has to escape in order to cross from one piece of the sample space into its complement. The benchmarks obtained by comparing escape times from an ε-relaxed Böröczky configurations may thus reflect the relative merits of sampling algorithms.

4.2 Böröczky Packings and Local MCMC: Fundamental Aspects

We now discuss fundamental aspects of the present work, namely the question of the irreducibility of local hard-sphere Markov chains and the connection with non-local MCMC algorithms (see Sect. 4.2.1), as well as regularization of Böröczky packings and ε-relaxed Böröczky configurations in the NPT ensemble (see Sect. 4.2.2).

4.2.1 Irreducibility of Local and Non-local Hard-Disk MCMC

Strictly speaking, ECMC can be irreducible only if $\Omega \setminus \mathcal{B}$ is connected, where $\mathcal{B}$ is a suitably defined space of locally stable configurations. Packings in $\mathcal{B}$ (a space of low dimension) are certainly invariant under any version of the ECMC algorithm, so that they cannot evolve towards other samples in Ω. Connectivity in $\Omega \setminus \mathcal{B}$ would at least assure that this space can be sampled. In addition it appears necessary to guarantee that a well-behaved initial configuration cannot evolve towards an ε-environment around $\mathcal{B}$ (e.g., the space $\mathcal{B}_\varepsilon$ of ε-relaxed Böröczky configurations that makes up a finite portion of Ω) or to gridlocks with diverging event rates. These properties appear not clearly established for finite densities η and for large N. In other models, for example the Ising model of statistical physics, irreducibility can be proven for any N.

These unresolved mathematical questions concerning irreducibility do not shed doubt on the practical usefulness of MCMC for particle systems. First, the concept of local stability is restricted to hard disks and hard spheres (that is, to potentials that are either zero or infinite). The phase diagram of soft-disk models can be continuously connected to the hard-disk case [30]. For soft disks, irreducibility is trivial, but the sampling speed of algorithms remains

crucial. Second, in applications, one may change the thermodynamic ensemble. In the NPT ensemble, the central simulation box fluctuates in size and can become arbitrarily large. In this ensemble, irreducibility follows from the fact that large enough simulation boxes are free of steric constraints. Again, the question of mixing and correlation time scales is primordial. Third, practical simulations that require some degree of irreducibility are always performed under conditions where the simulation box houses a number of effectively independent copies of the system. This excludes the crystalline or solid phases. Monte Carlo simulations of such phases are more empirical in nature. They require a careful choice of initial states, and are then not expected to visit the entire sample space during their time evolution. Fundamental quantitative results can nevertheless be obtained [31].

In this work, we concentrate on local MCMC algorithms, because global-move algorithms, as the cluster algorithms in spin systems, rely on *a priori* probabilities for many-particle moves that appear too complicated. Also, global single-particle moves are related to the single-particle insertion probabilities, in other words to fugacities (the exponential of the negative chemical potential) that are prohibitively small. At lower (finite) densities, however, placing at each time step a randomly chosen disk at a random position inside the box corresponds to the Metropolis algorithm of Sect. 3.1.1 with a square-shaped displacement set and a range $\delta = L/2$. This non-local algorithm easily escapes from a Böröczky packing. Moreover, it is proven to mix in $\mathcal{O}(N \log N)$ steps at densities $\eta < 1/6$ [8, 32] (see also [33]), a result that implies that the liquid phase in the hard-disk system extends at least to the density $\eta = 1/6$ [8]. The density bound for the algorithm (which yields a bound for the stability of the liquid phase) is much smaller than the empirical density bound for the liquid phase, at $\eta \simeq 0.70$. At this higher density, the global-move Metropolis algorithm and the more general hard-disk cluster algorithm [34] are almost totally stuck. For applications, we imagine structures resembling ε-relaxed Böröczky configurations to be backbones of configurations at high density, where global moves cannot be used.

4.2.2 Böröczky Packings and the *NPT* Ensemble

The concepts of packings and of local and collective stability make sense only in the NVT ensemble, that is, for a constant number of particles and for a simulation box with fixed shape and volume (the temperature $T = 1/\beta$ that appears in NVT plays no role in hard-disk systems [14]). In the NPT ensemble, the pressure P is constant, and the size of the simulation box may vary. The equivalence of the two ensembles is proven [35] for large N, so that the choice of ensemble is more a question of convenience than of necessity. As we will see, in the NPT ensemble, tiny relaxation parameters (as $\varepsilon = 10^{-29}$ in Fig. 6) are instantly relaxed to $\varepsilon \sim 10^{-3}$ for normal pressures and system sizes.

To change the volume at constant pressure, one may, among others, proceed to "rift volume changes" (see [36, Sect. VI]) or else to homothetic transformations of the central simulation box. We discuss this second approach (see [14, Sect. 2.3.4]), where the disk positions (but not the radii) are rescaled by the box size L as:

$$\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N) \to \boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_N) \quad \text{with } \boldsymbol{\alpha}_i = \mathbf{x}_i/L. \tag{15}$$

Each configuration is then specified by an $\boldsymbol{\alpha}$ vector in the $2N$-dimensional periodic unit square and an associated volume $V = L^2$, which must satisfy $V \geq V_{\text{cut}}(\boldsymbol{\alpha})$. A classic MCMC algorithm [37] directly samples the volume at fixed $\boldsymbol{\alpha}$ from a gamma distribution above $V_{\text{cut}}(\boldsymbol{\alpha})$, below which $(\boldsymbol{\alpha}, V)$ ceases to represent a valid hard-disk configuration [14, Eq. (2.19)]. Typical sample volumes are characterized by $\beta P(V - V_{\text{cut}}) \sim 1$, and with

$V = (L_{\text{cut}} + \Delta L)^2$, it follows that

$$\frac{\Delta L}{L} \sim \varepsilon \sim \frac{1}{\beta P V_{\text{cut}}} \quad \text{(at fixed } \boldsymbol{\alpha}\text{)}. \tag{16}$$

This equation illustrates that a packing, with $\varepsilon \to 0$, is realized as a typical configuration only in the limit $\beta P \to \infty$. For the Böröczky packings of Fig. 1, we have $L \simeq 20$, and a typical value for the pressure for hard-disk systems is $\beta P \sim 1$, which results in $\varepsilon \sim 10^{-3}$. In the NPT ensemble, as a consequence, escape times from a packing naturally correspond to a relaxation parameter $\varepsilon \sim 1/(\beta P V)$, in our example to $t_{\text{esc}}(\varepsilon \sim 10^{-3})$, which is $\mathcal{O}(1)$.

The above NPT algorithm combines constant-volume NVT-type moves of $\boldsymbol{\alpha}$ with the mentioned direct-sampling moves of V at fixed $\boldsymbol{\alpha}$. In practice, however, NPT calculations are rarely performed in hard-disk systems [38, 39]. This is because, as discussed in Eq. (16), the expected single-move displacement in volume at fixed $\boldsymbol{\alpha}$ is $\Delta V \sim 1/(\beta P)$, so that $\Delta V/V \sim 1/N$ (because $N \sim V$ and $\beta P \sim 1$). The fluctuations of the equilibrium volume V^{eq} (averaged over $\boldsymbol{\alpha}$) scale as $\sqrt{V^{\text{eq}}}$, which implies $\Delta V^{\text{eq}}/V^{\text{eq}} \sim 1/\sqrt{N}$. The volume-sampling algorithm requires $\sim N$ single updates of the volume to go from the $1/N$ scale of volume fluctuations at fixed $\boldsymbol{\alpha}$ to the $1/\sqrt{N}$ scale of the fluctuations of V^{eq} at equilibrium. This multiplies with the number of steps to decorrelate at a given volume. In practice, it has proven more successful to perform single NVT simulations, but to restrict them to physical parameters where the central simulation box houses a finite number of effectively independent systems mimicking constant-pressure configurations.

5 Conclusion

Building on an early breakthrough by Böröczky, we have studied in this work locally stable hard-disk packings. Böröczky packings are sparse, with arbitrarily small densities for large numbers N of disks. We constructed different types of these packings to arbitrary precision for finite N, namely Böröczky packings with the original Böröczky core [13] and those with the Kahle core [17]. In addition to the core and the number k of layers, Böröczky packings are defined by the convex polygonal chain which bounds their branches. We constructed Böröczky packings in a continuous range of densities, and made our software implementation of the construction openly accessible. Böröczky packings are locally, but not collectively stable. Using singular-value decomposition (in an implementation that is included in our open-source software) we explicitly exposed the unstable collective modes. We furthermore reduced the radius of Böröczky packings slightly, and determined the escape times from ε-relaxed Böröczky configurations as a function of the parameter ε for a number of local MCMC algorithms, including several variants of ECMC, arbitrary-precision implementations of which are also made openly available. Although the algorithms depart from each other in seemingly insignificant details only, we witnessed widely different escape times, ranging from $1/\varepsilon$ to $\log(1/\varepsilon)$. Our theory suggested that the significant speedup of some of the algorithms is rooted in their event-driven nature coupled to their lack of an intrinsic scale. We noted that the space of ε-relaxed Böröczky configurations is a finite portion of the sample space, and that a given computer-time budget implies such a finite fraction of sample space that is practically excluded in local MCMC at finite N. Here, the excluded volume only vanishes in the limit of infinite CPU time. More generally, connectedness of the hard-disk sample space is not proven. We pointed to the importance of statistical ensembles to reconcile the possible loss of irreducibility with the proven practical usefulness of local hard-disk MCMC algorithms. Although Böröczky packings or ε-relaxed Böröczky configurations are sparse, they could

form the locally stable (or almost locally stable) backbones of hard-disk configurations at the much higher density which are of practical interest.

We expect the observed differences in escape times to carry over to real-world ECMC implementations. Qualitatively similar performance differences were already observed in autocorrelation times of hard-disk dipoles [25]. In statistical mechanics, bottlenecks and escape times possibly play an important role in polymer physics and complex molecular systems and some of the algorithms studied here may find useful applications. Escape times may also play an important role in the study of glasses and in granular matter, where the high or even infinite pressures favor local configurations that resemble the mutually blocked disks in the ε-relaxed Böröczky configurations. We finally point out that the very concept of locally stable packings naturally extends to higher dimensions.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data Availability This work is based on computer programs that are all publicly available (see Appendix B). Data will also be made available on reasonable request.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

A Escape Times, Resamplings and Gridlocks

In this appendix we collect a number of numerical results that support statements made in the main text.

A.1 Critical Maximum Nearest-Neighbor Distance

In the escape time t_{esc} of Eq. (10), the parameter γ sets the critical maximum nearest-neighbor distance $d(t)$ for the escape from an ε-relaxed Böröczky configurations. In Sect. 3.2, we use $\gamma = 0.1$ which corresponds to a 10 %-increase of the initial value $d(t = 0) = 2$. Using the alternative value $\gamma = 0.025$, we find that the escape time of straight ECMC again varies algebraically as $t_{\text{esc}} \sim \varepsilon^{-2/3}$ and, for forward, reflective, and Newtonian ECMC we again find $t_{\text{esc}} \sim \ln(1/\varepsilon)$ (see Fig. 7). Our conclusions thus appear robust with respect to the value of γ.

A.2 Escape Times with Resamplings

The reflective, forward and Newtonian variants of ECMC, at a difference of straight ECMC, appear to not always require resampling. In the main text, we therefore use $\tau_{\text{chain}} = \infty$, which, given our discussion in Sect. 3.1, is appropriate. Moreover, resamplings after chain times τ_{chain} considerably deteriorate the escape time for all three variants (see Fig. 8). This again illustrates the power of lifted Markov chains, in which the proposed moves are correlated over long Monte-Carlo times.

A.3 Gridlock Rates with Different Numerical Precisions

The straight, reflective, and Newtonian variants of ECMC feature deterministic collision rules, and they may run into gridlocks if started from ε-relaxed Böröczky configurations for very small ε (see Sect. 3.2.4). In a gridlock, the active-disk label loops through a subset of

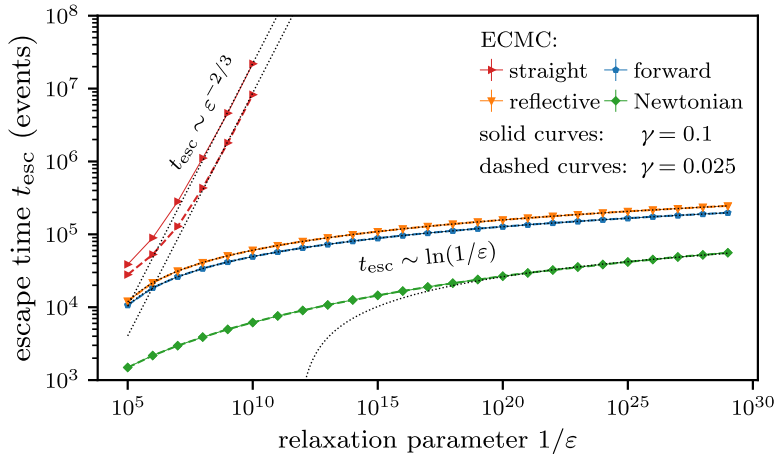

Fig. 7 Median escape times t_{esc} from $k = 5$ ε-relaxed Böröczky configurations (Kahle core and convex geometric chain $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.7$, $N = 96$ disks) for ECMC algorithms. Solid curves use $\gamma = 0.1$ for the definition of t_{esc} (as in Sect. 3.2), dashed curves use $\gamma = 0.025$ (see Eq. (10)). For both values of γ, straight ECMC with optimized chain time τ_{chain} shows algebraic scaling with identical exponents, whereas forward, reflective, and Newtonian ECMC scale logarithmically. Error bars are smaller than the marker sizes

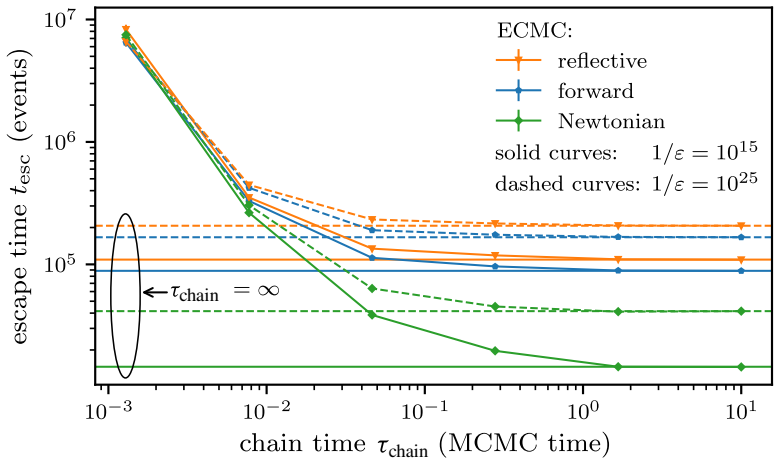

Fig. 8 Median escape times t_{esc} from $k = 5$ ε-relaxed Böröczky configurations (Kahle core and convex geometric chain $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.7$, $N = 96$ disks) for forward, reflective, and Newtonian ECMC *vs.* chain time τ_{chain} for two different relaxation parameters ε. Horizontal lines indicate the escape times without any resamplings. Error bars are smaller than the marker sizes

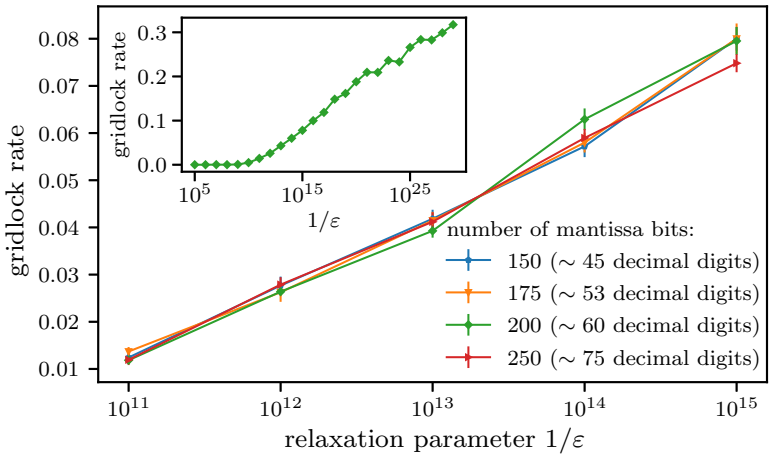

Fig. 9 Gridlock rate of Newtonian ECMC simulations with different numerical precisions starting from a $k = 5$ ε-relaxed Böröczky configurations (Böröczky core and convex geometric chain $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.7$, $N = 112$ disks). The inset shows the gridlock rate as a function of $1/\varepsilon$ for the Newtonian ECMC simulations with 200 mantissa bits that were used to measure the escape times in Fig. 6b

the N disks which are in contact. The event rate diverges, and so does the CPU time spent in the gridlock. The Monte-Carlo time, however, stands still. Newtonian ECMC starting from $k = 5$ ε-relaxed Böröczky configurations with the Böröczky core appears particularly prone to gridlocks.

It remains an open question whether gridlocks are a numerical artifact related to the finite-precision computer arithmetic. In our arbitrary-precision `BigBoro` software, the number of mantissa bits (in base 2) can be set freely. We have studied the gridlock rate of Newtonian ECMC (the fraction of simulations that run into gridlock) for the problematic $k = 5$ ε-relaxed Böröczky configurations (using the convex geometric chain $\mathcal{A}^{\text{geo}}$ with attenuation parameter $\phi = 0.7$) with the Böröczky core, and observed no clear influence of the numerical precision. It thus appears unlikely that gridlocks are a precision issue (see Fig. 9).

B `BigBoro` Software Package: Outline, License, Access

The `BigBoro` software package is published as an open-source project under the GNU GPLv3 license. It is available on GitHub as part of the `JeLLyFysh` organization.[1] The software package consists of three parts: First, the arbitrary-precision Python script `construct_packing.py` constructs finite-N Böröczky packings of hard disks in a periodic square box. Second, the Python script `collective_escape_modes.py` computes collective infinitesimal displacements of hard disks in a packing that result in an escape. Third, the arbitrary-precision Go application `go-hard-disks` performs hard-disk ECMC simulations that may start from ε-relaxed Böröczky configurations derived from Böröczky packings.

B.1 Python Script `construct_packing.py`

The arbitrary-precision Python script `construct_packing.py` implements the construction of Böröczky packings. It allows for the Böröczky or Kahle cores (see Sect. 2.1.1), and connects them to branches with a finite number of layers (see Sect. 2.1.3). The con-

[1] The url of repository is https://github.com/jellyfysh/BigBoro.

vex geometric chain $\mathcal{A}^{\text{geo}}$ with different attenuation parameters ϕ, and the convex circular chain $\mathcal{A}^{\text{circ}}$ are implemented (see Sect. 2.2.1). The core, the number of layers, and the convex polygonal chain are specified using command-line arguments. The construction of the Böröczky packings uses arbitrary-precision decimal floating-point arithmetic. Two additional command-line options specify the number of decimal digits, and the precision of the bisection search for the value $g_2^<$ that renders the Böröczky packing compatible with periodic boundary conditions (see Sect. 2.1.3). The final configuration and its parameters (as for example the system length) are stored in a human-readable format in a specified output file.

The `example_packings` directory of `BigBoro` contains several Böröczky packings in corresponding subdirectories (as for example `kahle_geometric_5`). The headers of these files contain the values of the command-line arguments for `construct_packing.py`. A plot of each example configuration is provided. The different packings in `kahle_geometric_5` and `boro_geometric_5` (see Fig. 1) were used in this work. Although the bisection search for the construction of the Böröczky packing usually requires an increased precision, the high-precision packings with small enough number of layers may be used as input for standard double-precision applications. For simplicity and improved readability, we provide `packing_double.txt` files that store the configurations in double precision, where applicable.

B.2 Python Script `collective_escape_modes.py`

The double-precision Python script `collective_escape_modes.py` identifies the orthonormal basis vectors of the escape matrix $\mathcal{M}^{\text{esc}}$ from a packing $\mathbf{x}$ (see Eq. (6)) that have zero singular values. Afterwards, these modes are classified using the cost function in Eq. (8). The resulting basis vectors $\boldsymbol{\Delta}_a$ form the solution space for $2N$-dimensional displacements $\boldsymbol{\Delta} = \{\Delta_1^x, \Delta_1^y, \Delta_2^x, \Delta_2^y, \ldots\}$ that have a vanishing first-order term in Eq. (5) and thus for collective infinitesimal displacements $\boldsymbol{\Delta}$ of all disks that escape from the packing. The basis vectors $\boldsymbol{\Delta}_a$ are stored in a human-readable output file, and optionally represented as in Fig. 2. The input filename of the packing, and the output filename for the collective escape modes are specified in command-line arguments. Further optional arguments specify the filename for the plots of the escape modes, and the system length of the central simulation box (that is unnecessary for packings generated by the Python script `construct_packing.py` in which case the system length is parsed from the packing file).

B.3 Go Application `go-hard-disks`

The Go application `go-hard-disks` relies on a cell-occupancy system for the efficient simulations of large-N hard-disk systems using several variants of the ECMC algorithm. Straight, reflective, forward, and Newtonian ECMC are implemented. After each sampling interval, it samples the maximum nearest-neighbor distance $d(t)$ (see Eq. (9)). All computations use a fixed number of mantissa bits (in base 2) that may exceed the usual 24 or 53 bits for single- or double-precision floating-point values. The ECMC variant, its parameters (as for example the sampling time or chain time), and further specifications (the number of mantissa bits, the cell specifications, the filename for the initial configuration, etc.) are again set using command-line arguments.

References

1. 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 (1953). https://doi.org/10.1063/1.1699114
2. Alder, B.J., Wainwright, T.E.: Phase transition for a hard sphere system. J. Chem. Phys. **27**, 1208 (1957). https://doi.org/10.1063/1.1743957
3. Alder, B.J., Wainwright, T.E.: Phase transition in elastic disks. Phys. Rev. **127**, 359 (1962). https://doi.org/10.1103/PhysRev.127.359
4. Bernard, E.P., Krauth, W.: Two-step melting in two dimensions: first-order liquid-hexatic transition. Phys. Rev. Lett. **107**, 155704 (2011). https://doi.org/10.1103/PhysRevLett.107.155704
5. Bernard, E.P., Krauth, W., Wilson, D.B.: Event-chain Monte Carlo algorithms for hard-sphere systems. Phys. Rev. E **80**, 056704 (2009). https://doi.org/10.1103/PhysRevE.80.056704
6. Krauth, W.: Event-chain Monte Carlo: foundations, applications, and prospects. Front. Phys. **9**, 229 (2021). https://doi.org/10.3389/fphy.2021.663457
7. Lebowitz, J.L., Penrose, O.: Convergence of virial expansions. J. Math. Phys. **5**, 841 (1964). https://doi.org/10.1063/1.1704186
8. Helmuth, T., Perkins, W., Petti, S.: Correlation decay for hard spheres via Markov chains (2020). https://arxiv.org/abs/2001.05323
9. Fejes, L.: Über einen geometrischen Satz. Math. Zeitschrift **46**, 83 (1940)
10. Conway, J.H., Sloane, N.J.A.: Sphere Packings, Lattices and Groups. Springer, New York (1999)
11. Torquato, S., Stillinger, F.H.: Jammed hard-particle packings: from Kepler to Bernal and beyond. Rev. Mod. Phys. **82**, 2633 (2010). https://doi.org/10.1103/RevModPhys.82.2633
12. Donev, A., Torquato, S., Stillinger, F., Connelly, R.: Jamming in hard sphere and disk packings. J. Appl. Phys. **95**, 989 (2004). https://doi.org/10.1063/1.1633647
13. Böröczky, K.: Über stabile Kreis- und Kugelsysteme. Ann. Univ. Sci. Budapest. Eötvös Sect. Math. **7**, 79 (1964)
14. Krauth, W.: Statistical Mechanics: Algorithms and Computations. Oxford University Press, Oxford (2006)
15. Michel, M., Durmus, A., Sénécal, S.: Forward event-chain Monte Carlo: fast sampling by randomness control in irreversible Markov chains. J. Comput. Graph. Stat. **29**, 689 (2020). https://doi.org/10.1080/10618600.2020.1750417
16. Klement, M., Engel, M.: Efficient equilibration of hard spheres with Newtonian event chains. J. Chem. Phys. **150**, 174108 (2019). https://doi.org/10.1063/1.5090882
17. Kahle, M.: Sparse locally-jammed disk packings. Ann. Comb. **16**, 773 (2012). https://doi.org/10.1007/s00026-012-0159-0
18. Pach, J., Sharir, M.: Combinatorial Geometry and Its Algorithmic Applications, Mathematical Surveys and Monographs, vol. 152. American Mathematical Society, Providence (2009)
19. Diaconis, P., Lebeau, G., Michel, L.: Geometric analysis for the Metropolis algorithm on Lipschitz domains. Invent. Math. **185**, 239 (2011). https://doi.org/10.1007/s00222-010-0303-6
20. Baryshnikov, Y., Bubenik, P., Kahle, M.: Min-type Morse theory for configuration spaces of hard spheres. Int. Math. Res. Not. **2014**, 2577 (2013). https://doi.org/10.1093/imrn/rnt012
21. Engel, M., Anderson, J.A., Glotzer, S.C., Isobe, M., Bernard, E.P., Krauth, W.: Hard-disk equation of state: first-order liquid-hexatic transition in two dimensions with three simulation methods. Phys. Rev. E **87**, 042134 (2013). https://doi.org/10.1103/PhysRevE.87.042134
22. Kapfer, S.C., Krauth, W.: Sampling from a polytope and hard-disk Monte Carlo. J. Phys. Conf. Ser. **454**, 012031 (2013). https://doi.org/10.1088/1742-6596/454/1/012031
23. Li, B., Todo, S., Maggs, A., Krauth, W.: Multithreaded event-chain Monte Carlo with local times. Comput. Phys. Commun. **261**, 107702 (2021). https://doi.org/10.1016/j.cpc.2020.107702
24. Bouchard-Côté, A., Vollmer, S.J., Doucet, A.: The bouncy particle sampler: a nonreversible rejection-free Markov chain Monte Carlo method. J. Am. Stat. Assoc. **113**, 855 (2018). https://doi.org/10.1080/01621459.2017.1294075
25. Höllmer, P., Maggs, A.C., Krauth, W.: Hard-disk dipoles and non-reversible Markov chains. J. Chem. Phys. **156**, 084108 (2022). https://doi.org/10.1063/5.0080101
26. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov Chains and Mixing Times. American Mathematical Society, Providence (2008)
27. Weigel, R.F.B.: Equilibration of orientational order in hard disks via arcuate event-chain Monte Carlo. https://theorie1.physik.uni-erlangen.de/research/theses/2018-ma-roweigel.html. Master thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg (2018)
28. Diaconis, P., Holmes, S., Neal, R.M.: Analysis of a nonreversible Markov chain sampler. Ann. Appl. Probab. **10**, 726 (2000). https://doi.org/10.1214/aoap/1019487508

29. Chen, F., Lovász, L., Pak, I.: Lifting Markov chains to speed up mixing. In: Proceedings of the 17th Annual ACM Symposium on Theory of Computing, p. 275 (1999)
30. Kapfer, S.C., Krauth, W.: Two-dimensional melting: from liquid-hexatic coexistence to continuous transitions. Phys. Rev. Lett. **114**, 035702 (2015). https://doi.org/10.1103/PhysRevLett.114.035702
31. Bolhuis, P.G., Frenkel, D., Mau, S.C., Huse, D.A.: Entropy difference between crystal phases. Nature **388**, 235 (1997). https://doi.org/10.1038/40779
32. Kannan, R., Mahoney, M.W., Montenegro, R.: Rapid mixing of several Markov chains for a hard-core model. In: Proc. 14th Annual ISAAC, Lecture Notes in Computer Science, pp. 663–675. Springer, Berlin (2003)
33. Bernard, E.P., Chanal, C., Krauth, W.: Damage spreading and coupling in Markov chains. EPL **92**, 60004 (2010). https://doi.org/10.1209/0295-5075/92/60004
34. Dress, C., Krauth, W.: Cluster algorithm for hard spheres and related systems. J. Phys. A **28**, L597 (1995). https://doi.org/10.1088/0305-4470/28/23/001
35. Ruelle, D.: Statistical Mechanics: Rigorous Results. World Scientific, Singapore (1999)
36. Michel, M., Kapfer, S.C., Krauth, W.: Generalized event-chain Monte Carlo: constructing rejection-free global-balance algorithms from infinitesimal steps. J. Chem. Phys. **140**, 054116 (2014). https://doi.org/10.1063/1.4863991
37. Wood, W.W.: Monte Carlo calculations for hard disks in the isothermal-isobaric ensemble. J. Chem. Phys. **48**, 415 (1968). https://doi.org/10.1063/1.1667938
38. Wood, W.W.: NpT-ensemble Monte Carlo calculations for the hard-disk fluid. J. Chem. Phys. **52**, 729 (1970). https://doi.org/10.1063/1.1673047
39. Lee, J., Strandburg, K.J.: First-order melting transition of the hard-disk system. Phys. Rev. B **46**, 11190 (1992). https://doi.org/10.1103/physrevb.46.11190

Publication 3: Hard-disk pressure computations—a historic perspective

The Journal of Chemical Physics

Hard-disk pressure computations—a historic perspective

Cite as: J. Chem. Phys. **157**, 234111 (2022); https://doi.org/10.1063/5.0126437
Submitted: 15 September 2022 • Accepted: 21 November 2022 • Published Online: 20 December 2022

Botao Li, Yoshihiko Nishikawa, Philipp Höllmer, et al.

The Journal of Chemical Physics ARTICLE scitation.org/journal/jcp

Hard-disk pressure computations—a historic perspective

Cite as: J. Chem. Phys. 157, 234111 (2022); doi: 10.1063/5.0126437
Submitted: 15 September 2022 • Accepted: 21 November 2022 •
Published Online: 20 December 2022

Botao Li,[1] Yoshihiko Nishikawa,[2] Philipp Höllmer,[3] Louis Carillo,[1] A. C. Maggs,[4] and Werner Krauth[1,a)]

AFFILIATIONS

[1] Laboratoire de Physique de l'Ecole normale supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université de Paris Cité, Paris, France
[2] Graduate School of Information Sciences, Tohoku University, Sendai 980-8579, Japan
[3] Physikalisches Institut and Bethe Center for Theoretical Physics, University of Bonn, Nussallee 12, 53115 Bonn, Germany
[4] CNRS Gulliver, ESPCI Paris, Université PSL, 10 rue Vauquelin, 75005 Paris, France

[a)] Author to whom correspondence should be addressed: werner.krauth@ens.fr

ABSTRACT

We discuss pressure computations for the hard-disk model performed since 1953 and compare them to the results that we obtain with a powerful event-chain Monte Carlo and a massively parallel Metropolis algorithm. Like other simple models in the sciences, such as the *Drosophila* model of biology, the hard-disk model has needed monumental efforts to be understood. In particular, we argue that the difficulty of estimating the pressure has not been fully realized in the decades-long controversy over the hard-disk phase-transition scenario. We present the physics of the hard-disk model, the definition of the pressure and its unbiased estimators, several of which are new. We further treat different sampling algorithms and crucial criteria for bounding mixing times in the absence of analytical predictions. Our definite results for the pressure, for up to one million disks, may serve as benchmarks for future sampling algorithms. A synopsis of hard-disk pressure data as well as different versions of the sampling algorithms and pressure estimators are made available in an open-source repository.

I. INTRODUCTION

In fundamental physics, the most detailed descriptions of physical reality are not always the best. In our quantum-mechanical world, many phenomena are, indeed, described classically, without the elaborate machinery of wavefunctions and density matrices. The exact thermodynamic singularities of helium, a quantum liquid, at the famous lambda transition[1] are, for example, obtained by a seemingly unrelated model of classical two-component spins[2–5] on a three-dimensional lattice rather than by some quantum-mechanical representation of all atoms in the continuum.[6] Renormalization-group theory[7] guarantees that the simple classical spin model exactly describes experiments in the quantum liquid.[8,9] The universality of simple models is also found in other sciences. In biology, the study of the fruit fly *Drosophila* has gradually evolved from a subject of entomology, the science of insects, to a parable for higher animals, where it allows one to appreciate gene damage[10] from radiation. In recent decades, it was, moreover, understood that many of the genes of *Drosophila* have exactly the same function as genes in vertebrates, including humans. In physics as in biology, "(t)his remarkable conservation came as a big surprise. It had been neither predicted nor expected,"[11] to cite a Nobel-prize winner.

Paradoxically, even simple models in science, those stripped to their bare bones, can take monumental efforts and decades of research to be understood fully. This is the case for the *Drosophila* fly that entered research laboratories around 1905[12] and then gradually turned into a model organism.[13] It is also what happened to the simplest of all particle models, the hard-disk model, which is the focus of the present work. The model consists of N identical classical disks with positions inside a box and with velocities. Disks fly in straight-line trajectories, except when they collide with each other or with an enclosing wall. The elementary collision rules are those of elastic, friction-less, billiard balls. The hard-disk model caricatures two-dimensional fluids: it lacks the explicit interparticle attractions of more realistic descriptions, yet it shows almost all the interesting properties of general particle systems. Moreover, its observed phase

behavior[14] was understood in terms of topological phase transitions, just like classical continuous spin models in two dimensions.[15] The interpretation common to both cases was unsuspected from the conventional Landau theory of phase transitions.

Only few characteristics of the hard-disk model are known from rigorous mathematics. The first was proven by Daniel Bernoulli in 1738,[16] namely, that the temperature, which is linked to the mean-square velocity of the disks, plays no role in their spatial distribution. It was also proven[17,18] that the hard-disk model, as a dynamical system with positions and velocities evolving under Newton's laws, can be described statistically with positions that all have equal statistical weight. It is furthermore shown rigorously that, at small finite density, the model is fluid,[19,20] justifying analytic expansions developed in the 19th century[21] to link dilute hard disks to the ideal gas of non-interacting particles. For high densities, it was established that at close packing, the hard-disk model forms a hexagonal crystal.[22] However, for all densities below close packing, this crystalline structure is destroyed[23,24] by long-wavelength fluctuations. The invention of simulation methods, and their application to this very model of hard disks ever since the 1950s, was meant to overcome the scarcity of analytical results.

With its stripped-down interactions, the hard-disk model is, indeed, simple. In particular, the model lacks attractive forces that would pull the disks together. It is for this reason that, for a long time, hard disks and hard spheres (in three dimensions) were considered to be too simple to show a phase transition from a disordered fluid to a solid.[25,26] In two dimensions, furthermore, ordered phases were expected not to exist for theoretical reasons that were considered sufficiently solid to formally exclude any transition.[27] Initial computer simulations, in 1953, in the same publication that introduced the Metropolis algorithm,[28] accordingly found that "(t)here is no indication of a phase transition." It thus came as an enormous surprise[29,30] when, in 1962, Alder and Wainwright[14] identified a loop in the equation of state (see Fig. 1), suggesting[31] a phase transition between a fluid at low density and a solid (that was not supposed, at the time, to exist) at high density. This laid the ground work for the Kosterlitz–Thouless–Halperin–Nelson–Young theory of melting in two-dimensional particle systems.[32–34] Even after this

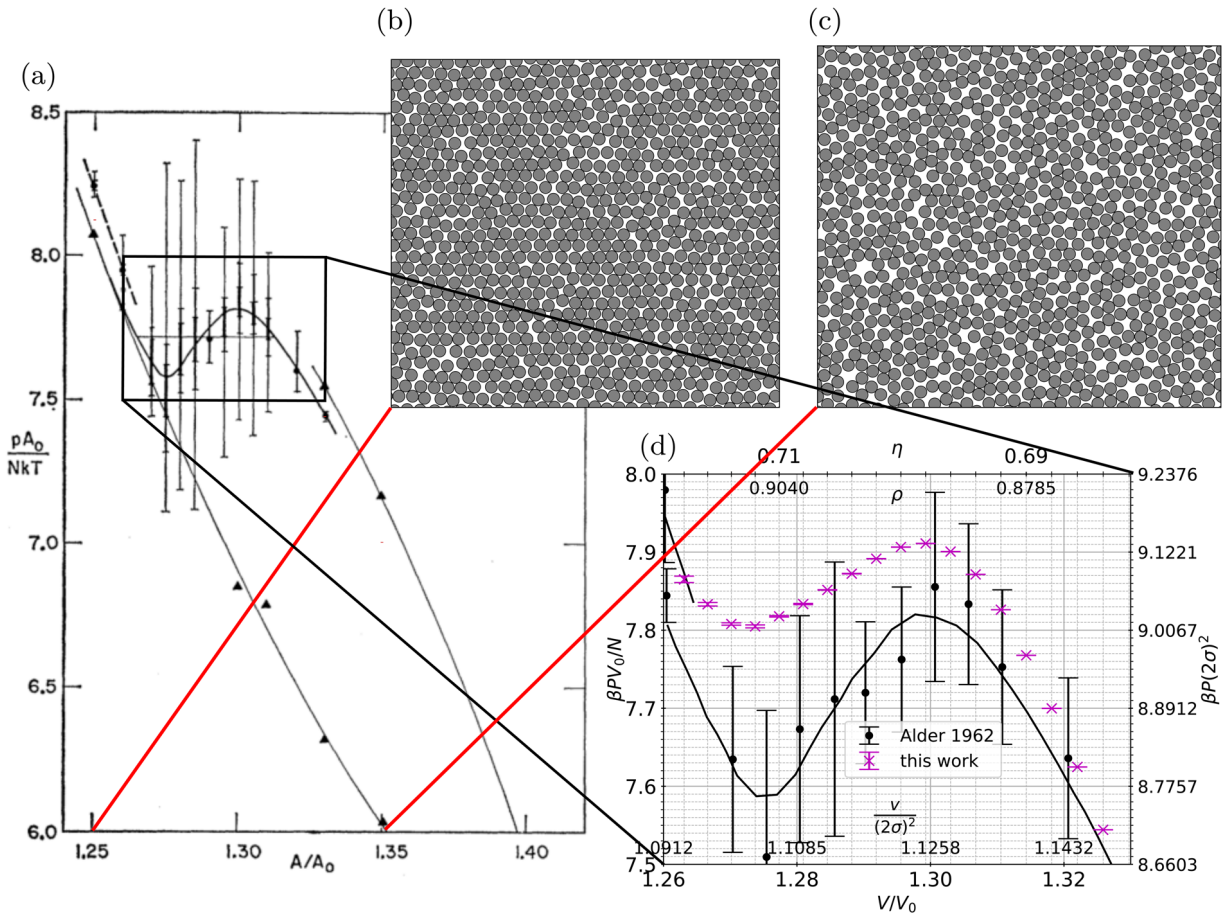

FIG. 1. Hard disks in a periodic box. (a) Equation of state (pressure vs volume) computed in 1962 by Alder and Wainwright.[14] (b) and (c) Samples of 870 disks at densities $\eta = 0.726$ and $\eta = 0.672$. (d) Pressures for $N = 870$ from (a) compared with ECMC results (this work) for aspect ratio $\alpha = (9 : 8\sqrt{3}/2)$ (*cf.* Fig. 10 for an analysis of convergence behavior).

important conceptual advance, the phase behavior and the equation of state of the hard-disk model remained controversial for another fifty years until an analysis[35] based on the powerful event-chain Monte Carlo (ECMC) algorithm[36] showed that hard disks melt through a first-order fluid–hexatic phase transition combined with a Kosterlitz–Thouless transition between the hexatic and the solid, thus proposing a new scenario.

In this work, we discuss the hard-disk model in a computational and historical context, concentrating on the pressure as a function of the volume. The reason for this restriction of scope on pressures rather than on phases is that the aforementioned 50-year controversy on the hard-disk phases takes its origin in difficulties in computing the pressure as the dependent variable in the equation of state. After a short introduction to the physics of the hard-disk model, we review the thermodynamic and kinematic pressure definitions and show that they are perfectly equivalent even for finite systems. Nevertheless, the pressure may be anisotropic and depend on the shape of the system, rather than being only a function of system volume. We discuss pressure estimators, with a focus on those that are unbiased at finite N and are convenient to use. We furthermore clarify that different sampling algorithms [molecular dynamics, reversible and non-reversible Markov-chain Monte Carlo (MCMC)] all rigorously sample the same equal-weight Boltzmann distribution of positions although the time scales for convergence can differ widely even for local algorithms and can reach years and even decades of computer time for moderate system sizes. This was not understood in many important historical contributions. With all this material in hand, we confront past results with massive computations performed for this work, thus providing definite high-precision pressure estimates for the hard-disk model that may serve as benchmarks for the future. With its rich phenomenology and its intractable mathematics, this simple model has become the *Drosophila* for particle systems and for two-dimensional phase transitions. It has served as a parable for difficult computing problems and as a platform for the development of MCMC and of molecular dynamics. This fascinating model has yet to be fully understood. We aim at providing a solid base for future work.

The plan of this work is as follows: In Sec. II, we discuss the fundamentals of the hard-disk model, from the definitions of densities and volumes to an overview of its physical properties. In Sec. III, we discuss sampling algorithms molecular dynamics and MCMC and pressure estimators for this model, concentrating on new developments. In Sec. IV, we digitize, discuss, and make available numerical computations of the equation of state performed since the paper by Metropolis *et al.*, in 1953, and superpose them with large-scale computations done for this work. Section V contains our conclusions. We also provide information on statistical analysis (Appendix A) and introduce to the `HistoricDisks` open-source software package (Appendix B), which collects pressure data since 1953 and implements sampling algorithms and pressure estimators that were used in this work.

II. HARD-DISK MODEL

The hard-disk model consists of N impenetrable, identical, classical disks of radius σ and mass m, which are perfectly elastic. Collisions are instantaneous; they cause no deformations and induce no rotations. Pair collisions conserve momentum and energy. Disks evolve in a fixed rectangular box of sides L_x and L_y [specified through the aspect ratio $\alpha = (L_x : L_y)$], which may have periodic boundary conditions ("periodic" box), or else hard-wall boundary conditions ("non-periodic" box). The two-dimensional volume (area) is $V = L_x \times L_y$. In the NVT ensemble that we consider here, N, L_x, and L_y are fixed. For the hard-disk system, the microcanonical NVE ensemble (of constant energy E) and the canonical NVT ensemble (of constant temperature T) are almost equivalent, and we connect the two throughout this work. In other ensembles, the box can be of varying dimensions,[37–39] and N might vary.[40] The disk i is described by the position of its center $\mathbf{x}_i = (x_i, y_i)$ and possibly by a velocity $\mathbf{v}_i = (v_{x,i}, v_{y,i})$. We denote the set of positions and velocities by $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ and $\mathbf{v} = \{\mathbf{v}_1, \ldots, \mathbf{v}_N\}$, respectively.

A. Basic definitions and properties

We now discuss additional characteristics of the hard-disk model and define its Newtonian dynamics. Furthermore, we summarize the physics of two-dimensional particle systems.

1. System definitions, basic properties

In the limit $N \to \infty$, the properties of a particle system with short-range interactions are independent of the boundary conditions. At finite N, in contrast, the bulk part of the free energy (that scales as V) cannot be separated from the surface term (that, in two dimensions, scales as $\sqrt{V}$). For example, a close-packed crystal of $N = 72 = 8 \times 9$ disks can fit into a periodic box of aspect ratio $\alpha = (9 : 8\sqrt{3}/2)$, whereas the maximum density for the same N in a box with $\alpha = (1 : \sqrt{3}/2)$ is slightly smaller (see Fig. 2). Evidently, the pressure in a box containing a finite number of disks depends on its aspect ratio.

The following conventions for the volume or its inverse, the density, have been commonly used in the literature. The first is the volume $V = L_x \times L_y$ normalized by that of a perfect hexagonal packing $V_0 = N(2\sigma)^2\sqrt{3}/2$ [as in Fig. 2(b)]. Second is the covering density η, the total volume of all disks normalized by the box volume V. Third is the reduced density ρ, the number N of disks of radius $\frac{1}{2}$ divided by the volume and, finally, the inverse normalized density $v/(2\sigma)^2$ with $v = (2\sigma)^2/\rho$. These quantities are related as follows:

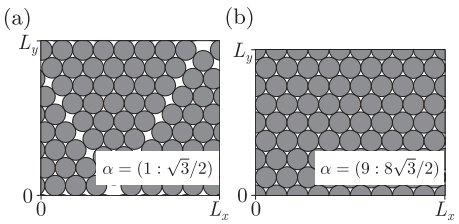

FIG. 2. Packings for $N = 72$ disks for different aspect ratios α. (a) Periodic box with $\alpha = (1 : \sqrt{3}/2)$, with conjectured optimal packing. (b) Periodic box with $\alpha = (9 : 8\sqrt{3}/2)$ at the close-packing density $\eta = \pi/(2\sqrt{3})$.

$$\begin{aligned}\frac{V}{V_0} &= \frac{\pi}{2\sqrt{3}}\frac{1}{\eta} = \frac{2}{\sqrt{3}}\frac{v}{(2\sigma)^2} \geq 1,\\ \eta &= \frac{\pi}{2\sqrt{3}}\frac{V_0}{V} = \frac{N}{V}\pi\sigma^2 \leq 0.907,\\ \rho &= \eta\frac{4}{\pi} = \frac{V_0}{V}\frac{2}{\sqrt{3}} = \frac{(2\sigma)^2}{v} \leq 1.155,\\ \frac{v}{(2\sigma)^2} &= \frac{\sqrt{3}}{2}\frac{V}{V_0} = \frac{\pi}{4}\frac{1}{\eta} = \frac{1}{\rho} \geq 0.866.\end{aligned} \tag{1}$$

We will re-plot published equations of state with all four units, thus simplifying the comparison of data.

In the hard-disk model, all configurations have zero potential energy, and the Newtonian time evolution conserves the kinetic energy. Pairs of disks collide such that, in their center-of-mass coordinate system, they rebound from their line of contact with conserved parallel and reversed perpendicular velocities.[41] At a wall collision, the velocity component of a disk parallel to the wall remains the same while the perpendicular velocity $v_{\text{wall}}^{\perp}$ is reversed. When, at the initial time $t = 0$, all velocities are rescaled by a factor γ, the entire trajectory transforms as

$$\{\mathbf{x}_1(t), \ldots, \mathbf{x}_N(t)\} \xrightarrow{\mathbf{v}_i \to \gamma \mathbf{v}_i \forall i} \left\{\mathbf{x}_1\left(\frac{t}{\gamma}\right), \ldots, \mathbf{x}_N\left(\frac{t}{\gamma}\right)\right\}. \tag{2}$$

This property of hard-sphere models was already noticed by Daniel Bernoulli.[16]

Statistically, during the Newtonian hard-disk time evolution, the sets of positions and of velocities are mutually independent. All positions $\mathbf{x}$ that correspond to N-disk configurations have the same statistical weight π with a Cartesian density measure and, in a non-periodic box, the velocities are randomly distributed on the surface of the hypersphere in the $2N$-dimensional space with radius $R_v = \sqrt{2E/m}$, where E is the conserved (microcanonical) energy of the NVE ensemble. The measure in the $4N$-dimensional sample space of positions and velocities is, thus,

$$\mathrm{d}\pi \propto \Theta(\mathbf{x})\mathrm{d}^N\mathbf{x}\,\mathrm{d}^N\mathbf{v}\,\delta\left(E - \sum_i m\mathbf{v}_i^2/2\right), \tag{3}$$

where $\Theta(\mathbf{x}) = 1$ for positions that correspond to hard-disk configurations and $\Theta(\mathbf{x}) = 0$ otherwise. In a periodic box, in addition, the two components of the total velocity and the position of the center of mass in the rest frame are conserved. For large N, where the ensembles are equivalent, the microcanonical energy per disk corresponds to $E/N = k_{\mathrm{B}}T = 1/\beta$, where k_{B} is the Boltzmann constant and T is the temperature of the canonical ensemble. The spatial part of the measure of Eq. (3) remains unchanged, and the uniform velocity distribution on the surface of the hypersphere in $2N$ dimensions implies that the one-particle, marginal distribution of velocity components becomes Gaussian,

$$\pi(v_{i,x}) \propto \exp\left(-\beta m v_{i,x}^2/2\right) \quad (N \to \infty) \tag{4}$$

(and analogously for $v_{i,y}$, see Ref. 41, Sec. 2.1.1).

The probability distribution of the velocity perpendicular to a wall $v_{\text{wall}}^{\perp}$ at the discrete wall-collision times (essentially the histogram of momentum transfers with the walls) differs from Eq. (3). For $N \to \infty$, this distribution is biased by a factor $|v_{\text{wall}}^{\perp}|$ with respect to the Maxwell distribution,

$$\pi(|v_{\text{wall}}^{\perp}|) \propto |v_{\text{wall}}^{\perp}| \exp\left[-\beta m (v_{\text{wall}}^{\perp})^2/2\right], \tag{5}$$

which has been described through the "Maxwell boundary condition" [see (Ref. 41, Sec. 2.3.1)]. For finite N, the same biasing factor appears. The distribution of the velocity perpendicular to a wall is derived from the surface element on the hypersphere of radius $R_v = \sqrt{v_1^2 + \cdots + v_n^2}$ in $n = 2N$ dimensions,

$$\mathrm{d}\Omega = R_v^{n-1} \sin^{n-2}\phi_1 \sin^{n-3}\phi_2 \cdots \sin\phi_{n-2}\mathrm{d}\phi_1 \cdots \mathrm{d}\phi_{n-1}, \tag{6}$$

where $\phi_1, \ldots, \phi_{n-2} \in [0, \pi]$ and $\phi_{n-1} \in [0, 2\pi]$, and where only $v_1 = R_v \cos\phi_1$ is expressed in terms of a single angle. It is thus convenient to identify v_1 with $v_{\text{wall}}^{\perp}$. The radius R_v of the hypersphere at the microcanonical energy $E = mR_v^2/2$ is related to the inverse temperature in the canonical ensemble as $R_v^2 = 2N/(m\beta)$. With the integrals

$$\begin{aligned}A &= \int_0^{\pi} \mathrm{d}\phi_1 |\cos\phi_1| \sin^{n-2}\phi_1 = \frac{2}{n-1},\\ B &= \int_0^{\pi} \mathrm{d}\phi_1 \sin^{n-2}\phi_1 = \sqrt{\pi}\frac{\Gamma[(n-1)/2]}{\Gamma(n/2)},\end{aligned} \tag{7}$$

this yields

$$\left\langle \frac{1}{|v_{\text{wall}}^{\perp}|} \right\rangle = \frac{1}{R_v}\frac{B}{A} = \frac{\sqrt{\pi}}{R_v}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \xrightarrow{N\to\infty} \sqrt{\frac{\pi m\beta}{2}}, \tag{8a}$$

$$\left\langle |v_{\text{wall}}^{\perp}| \right\rangle = R_v\frac{B}{2NA} = \frac{R_v\sqrt{\pi}}{2N}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \xrightarrow{N\to\infty} \sqrt{\frac{\pi}{2m\beta}}, \tag{8b}$$

where in the limit $N \to \infty$, the ratio of the Γ functions approaches $\sqrt{N}$. The relative perpendicular velocities $\Delta v_{ij}^{\perp}$ (the projection of the relative velocity $\mathbf{v}_i - \mathbf{v}_j$ perpendicular to the interface separating disks i and j at their collision) are, similarly,

$$\left\langle \frac{1}{|\Delta v_{ij}^{\perp}|} \right\rangle = \frac{\sqrt{2\pi}}{R_v}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \xrightarrow{N\to\infty} \sqrt{\pi m\beta}, \tag{9a}$$

$$\left\langle |\Delta v_{ij}^{\perp}| \right\rangle = \frac{R_v\sqrt{\pi}}{\sqrt{2}N}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \xrightarrow{N\to\infty} \sqrt{\frac{\pi}{m\beta}}. \tag{9b}$$

2. Pair correlations, entropic phase transition

In the hard-disk model, the Boltzmann weights are the same for all sets of disk positions, since there are no explicitly varying interactions. In consequence, the two possible fluid phases (namely, the gas and the liquid) are confounded. A purely entropic "depletion" interaction[42] between disks, nevertheless, arises from the presence of other disks, effectively driving phase transitions. The three phases of the hard-disk model are fluid (with exponential decays of the orientational and positional correlation functions), hexatic (with an algebraic decay of orientational and exponential decay of

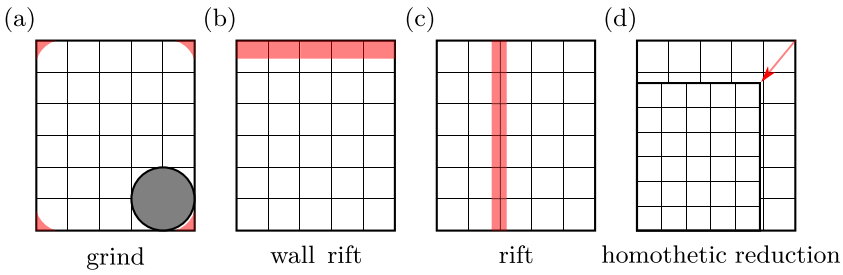

FIG. 3. Volume reductions for a non-periodic box. (a) Pathological corner-grind volume reduction that eliminates no samples for sufficiently large σ. (b) Horizontal wall rift used to estimate the pressure P_y. (c) Vertical rift used to estimate P_x. (d) Homothetic volume reduction.

positional correlations), and solid (with long-range orientational correlations and an algebraic decay of positional correlations). The hexatic and solid phases have only been identified through numerical simulations, and mathematical proofs of their existence are still lacking.

B. Hard-disk thermodynamics

In statistical mechanics, a homogeneous system (composed of, say, N particles in a fixed box) is described by an equation of state connecting two quantities, for example, the volume and the pressure. When for some volumes, a homogeneous phase may not exist, two (or exceptionally three) phases may coexist. We now link the definitions of the pressure from the thermodynamic and kinematic viewpoints and then discuss phase coexistence in finite systems.

1. Pressure, thermodynamic, and kinematic definitions

In statistical mechanics, the pressure P is given by the change in the free energy with the system volume,

$$\beta P = \frac{\partial \log Z}{\partial V} = \lim_{V' \to V} \frac{1}{V - V'} \frac{Z - Z'}{Z}, \tag{10}$$

with Z being the partition function and $Z' \equiv Z(V')$. For hard disks and related models, the rightmost fraction in Eq. (10) expresses the probability that a sample in the original box of volume V is eliminated in the box of reduced volume $V' < V$ [see Figs. 3(a)–3(c)]. In rift-pressure estimators,[43] the volume V of an $L_x \times L_y$ box is reduced by removing an infinitesimal vertical or horizontal slab (a "rift"), yielding the components P_x and P_y of the pressure. Rifts can be placed anywhere in the box, and one may even average any given hard-disk sample over all vertical or horizontal rift positions (see Subsection III C). We will also discuss homothetic pressure estimators, where all box dimensions and disk coordinates are scaled down by a common factor [see Fig. 3(d)]. Used for decades, they estimate the pressure $P = (P_x + P_y)/2$.

Besides the thermodynamic definition of the pressure, one can also define the kinematic pressure as the exchange of momentum with the enclosing walls. However, thermodynamic and kinematic pressures are rigorously identical already at finite N, and the corresponding estimators can be transformed into each another. This remains true for the pressure estimators built on the virial formalism that we also discuss.

2. Equation of state, phase coexistence

In the thermodynamic limit, the stability of matter is expressed through a positive compressibility $\kappa = -(\partial V/\partial P)/V$. For a finite system in the NVT ensemble, this is not generally true in a presence of a first-order phase transition, where two coexisting phases are separated by an interface with its own interface free energy.

In a periodic two-dimensional box for finite N, on increasing the density (decreasing the volume), a first-order phase transition first creates a bubble of the denser phase in the less dense phase (for hard disks: a hexatic bubble inside the fluid). The stabilization of this bubble requires an extra "Laplace" pressure corresponding to the surface tension, which renders the overall pressure non-monotonic with V.[31] At larger densities, the bubble of the minority phase transforms into a stripe that winds around the periodic box. In the stripe regime, the length of the two interfaces and, thus, the interface free energy do not change with V, so that the pressure remains approximately constant. Finally, one obtains a bubble of the less dense phase (the fluid) in the surrounding denser (hexatic) phase (see Fig. 4). In the NVT ensemble, phase coexistence takes place at certain specific volumes V/V_0 that do not correspond to densities $\eta = (N/V)\pi\sigma^2$ of a homogeneous stable phase for $N \to \infty$. Phase coexistence is absent in the NPT ensemble. The pressure is then the control variable, and the volume V/V_0 is discontinuous at the

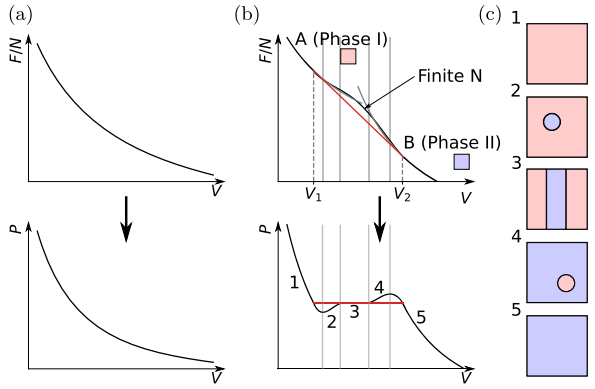

FIG. 4. First-order phase transition in the NVT ensemble. (a) Free energy with increasing second derivative and, thus, a monotonously decreasing pressure. (b) Free energy with—for the infinite system—metastable branches starting at volumes V_1 and V_2 and a non-monotonous equilibrium pressure P for finite N. (c) Sequence of five regimes in a finite two-dimensional periodic box, with pure (1, 5), bubble (2, 4), and stripe (3) phases.

transition, providing a simpler physical picture. However, in the *NPT* ensemble and its variants, sampling algorithms generally converge even more slowly than in the *NVT* ensemble.

The phase coexistence and the non-monotonous equation of state are genuine equilibrium features at finite N. Moreover, the spatially inhomogeneous phase-separated equilibrium state is reached from homogeneous initial configurations through a slow coarsening process, where an extensive number of bubbles coalesce into a single bubble or a single stripe [see Fig. 4(c)]. The coarsening dynamics depends on the sampling algorithm.

III. SAMPLING ALGORITHMS AND PRESSURE ESTIMATORS

In this section, we discuss sampling algorithms (molecular dynamics, MCMC) and pressure estimators for the hard-disk model. All algorithms are unbiased and correct to machine precision. For example, molecular dynamics implements the Newtonian time evolution of disks without discretizing time. The reversible Monte Carlo algorithms, from the historic Metropolis algorithm to the recent massively parallel Monte Carlo (MPMC) on graphics-card computers, satisfy the detailed-balance condition: the net equilibrium flow vanishes between any two samples. The non-reversible ECMC algorithms only satisfy the global-balance condition, and samples are taken in an extended "lifted" sample space.

Besides the sampling algorithms, we also discuss pressure estimators, including recent ones that overcome the limitations of the traditional approaches. Thermodynamic pressure estimators compute the probability with which a sample is eliminated as the box is reduced in size while kinematic pressure estimators determine the momentum fluxes at the walls or inside the box. Importantly, both types of estimators have the same expectation value (the pressure P or its components P_x or P_y), and they merely differ in their efficiency and ease of use. Even at finite N, there is, thus, no ambiguity in the definition of the pressure. The various estimators play a key role in the equation-of-state computations in Sec. IV. All algorithms and estimators are cross-validated for four disks in a non-periodic square box (see Appendix B 2 a). These tiny simulations illustrate the absence of any finite-N bias in the pressure estimators.

A. Event-driven molecular dynamics (EDMD)

Event-driven molecular dynamics (EDMD)[44] implements the Newtonian time evolution for the hard-disk model by stepping forward from one event (pair collision or wall collision) to the next. Between collisions, the disks move with constant velocities. Collisions of more than two disks or simultaneous collisions can be neglected. For large run times τ_{sim}, if the sample space is connected, molecular dynamics samples the equilibrium distribution of positions and velocities of Eq. (3).

1. Naive molecular-dynamics program

From a given set of positions and velocities for $N = 4$ disks in a non-periodic box, our naive EDMD code computes the minimum over all pair collision times for the $N(N-1)/2$ pairs of disks and over the wall collision times for the N disks. This minimum corresponds to a unique collision event (multiple overlaps appearing with finite-precision arithmetic can be treated in an ad-hoc fashion). The code then updates all the positions and the velocities of the colliding disks. This algorithm is of complexity $\mathcal{O}(N^2)$ per event. A related naive program in a periodic box (with arbitrary N) is used for cross-validation of other algorithms. Practical implementations of EDMD process the collisions through floating-point arithmetic. As the hard-disk dynamics is chaotic for almost all initial configurations, trajectories for different precision levels quickly diverge and only the statistical properties of the trajectories are believed to be correct. Naive EDMD provides proof of concept, and it can yield results for small N on modern computers. The EDMD code used by Alder and Wainwright already implemented sophisticated optimization strategies, for example, the tracking of neighbors, as required by the then available machines.[45]

2. Modern hard-disk molecular dynamics

The complexity of EDMD can be reduced from the naive $\mathcal{O}(N^2)$ per event scaling to $\mathcal{O}(\log N)$. This is because the collisions of a given disk must only be tracked with other disks in a local neighborhood [reducing by itself the complexity to $\mathcal{O}(N)$] and because the collision of two disks i and j only modifies the future collision times for pairs involving i or j (see Refs. 45–47). This algorithm keeps $\mathcal{O}(N)$ candidate events of which a finite number must be updated after each event. Using a heap data structure, this is of complexity $\mathcal{O}(\log N)$, while the retrieval of the shortest candidate event time (the next event time) is $\mathcal{O}(1)$. Although the update of the event times involves elaborate book-keeping, and although the processing of events according to collision rules is time-consuming, the EDMD algorithm is thus fairly efficient.[48,49] It has been successfully used for the hard-disk model up to intermediate sizes ($N \lesssim 256^2$ in the transition region). Open-source implementations of this algorithm are available.[50] The EDMD algorithm has not been successfully parallelized, despite some efforts in that direction.[51–53]

B. Hard-disk Markov-chain Monte Carlo

Hard-disk Monte-Carlo algorithms consider a sample space consisting of the N positions. Initial samples that are easy to construct are modified through reversible or non-reversible schemes. In the large-time limit, the equal-probability measure of the positions in Eq. (3) is reached.

1. Local hard-disk Metropolis algorithm

In the local hard-disk Metropolis algorithm,[28] at each time step, a small random displacement of a randomly chosen disk is accepted if the resulting sample is legal and is rejected otherwise. A move and its inverse are proposed with the same probability, so that the algorithm satisfies the detailed-balance condition with a constant equilibrium probability, and the net probability flows vanish.

The local Metropolis algorithm has been much used to obtain the hard-disk equation of state. On a modern single-core central-processing unit (CPU), this algorithm realizes roughly $\sim 10^{10}$ moves per hour. (For simplicity, we use "moves" for "proposed moves.") However, its convergence is very slow. In Sec. IV B 2, we will show evidence of mixing times[54] in excess of 10 years of CPU time for $\sim 10^6$ disks. The sequential variant of the local Metropolis algorithm updates the disk $i+1$ (identifying $N+1 \equiv 1$) at time $t+1$ after having updated disk i at time t. This non-reversible version runs slightly faster as it requires fewer random numbers per move, but the performance gain is minimal.

2. Massively parallel Monte Carlo (MPMC) algorithm

The MPMC algorithm generalizes the local Metropolis algorithm for implementation on graphical processing units (GPU).[55] It uses a four-color checkerboard of rectangular cells of sides larger than 2σ, which is superposed onto the periodic box and is compatible with the periodic boundary conditions. Cells of the same color are distant by more than 2σ. They are aligned with the $\hat{\mathbf{e}}_x$ and $\hat{\mathbf{e}}_y$ axes. The MPMC algorithm samples one of the four colors and then independently updates disks in all corresponding cells using the local Metropolis algorithm with the additional constraint that disks cannot leave their cells (see Fig. 5). After a certain time, the color is resampled. The checkerboard is frequently detached from the box, then randomly translated and repositioned, rendering the algorithm irreducible.

On a single NVIDIA GeForce RTX3090 GPU, our MPMC code reaches 2.1×10^{13} moves per hour, an order of magnitude more than an earlier implementation (Ref. 56, Table II). Repositioning the checkerboard is computationally cheap and is done often enough for the convergence time, measured in moves, to be only slightly larger than for the local Metropolis algorithm.

3. Hard-disk event-chain Monte Carlo (ECMC)

Hard-disk ECMC is a non-reversible continuous-time "lifted" Markov chain[36] in which—on a single processor—at each time t, a single active disk moves with constant velocity, while all the others are at rest. The identity and velocity of the active disk constitute additional "lifting variables" in an extended (lifted) sample space.[57–59] At a collision event, the active disk stops, and the target disk becomes active. The variants of ECMC differ in how the velocity is updated. In the "straight" variant of ECMC, the velocity of the active disks is maintained after a pair-collision event. It is usually chosen to be either in the $\pm\hat{\mathbf{e}}_x$ or the $\pm\hat{\mathbf{e}}_y$ direction, i.e., along one of the coordinate axes. At a wall-collision event, the velocity is flipped, for example, from $\pm\hat{\mathbf{e}}_x$ to $\mp\hat{\mathbf{e}}_x$. In addition, resampling events take place typically at equally spaced times separated by the run-time interval τ_{sim}. At such resamplings, a new active disk is sampled, and the new velocity is sampled from $\{\pm\hat{\mathbf{e}}_x, \pm\hat{\mathbf{e}}_y\}$. With periodic boundary conditions, the new velocity is sampled from $\{+\hat{\mathbf{e}}_x, +\hat{\mathbf{e}}_y\}$.

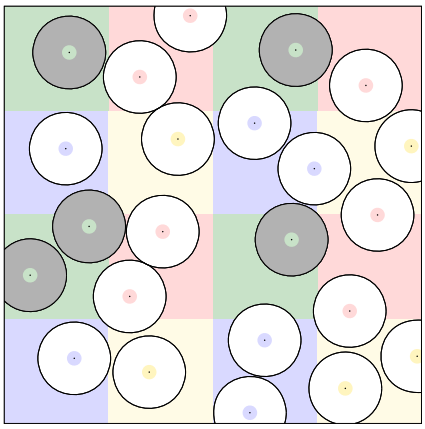

FIG. 5. Four-color checkerboard decomposition in a periodic box, with cells larger than 2σ. If the *green* color is chosen, highlighted disks may move, but cannot leave their cells. Disks in different green cells do not communicate.

Under conditions of irreducibility and aperiodicity, ECMC samples the equilibrium distribution of hard-disk positions with non-zero net probability flows. However, the hard-sphere ECMC and the hard-sphere local Metropolis algorithm are not strictly irreducible.[60] ECMC is much more powerful than the local Metropolis algorithm, and in Sec. IV B 2, we will evidence speedup factors of $\sim 10^3$. The `HistoricDisks` software package (see Appendix B) contains sample codes for straight ECMC, reflective ECMC,[36] forward ECMC,[61] and Newtonian ECMC.[62] Straight ECMC is fastest for the hard-disk model, and it was successfully parallelized.[63] The performance of straight ECMC is roughly of 10^{10} events (collisions) per hour on a modern single-core CPU. Its parallelized version reaches $\lesssim 10^{11}$ events per hour. This performance is currently limited by a hardware bandwidth bottleneck,[64] which will be overcome in the near future.

C. Hard-disk pressure estimators

As discussed in Subsection II B 1, the pressure describes, on one hand, the change in the free energy when the volume is reduced and, on the other hand, the time-averaged momentum exchange with the walls. In this subsection, we reduce the volume through rifts and rift averages and by uniformly shrinking the box. We also compute the momentum exchange directly and through a virial formula. Our motivation is two-fold. First, we obtain practical pressure estimators that we implement in our algorithms. Second, we discuss in detail that all the pressure estimators of Monte Carlo and of molecular dynamics compute the same object, and this even for finite systems. The decades-long discrepancies in the estimated pressures can thus not be traced to differences in their definitions.

1. Rifts and rift averages

In an $L_x \times L_y$ box, the volume may be reduced through a vertical "rift" $[x_r, x_r + \epsilon] \times [0, L_y]$ with disk positions transforming as

$$\{x, y\} \to \begin{cases} \{x, y\} & \text{if } x < x_r \\ \emptyset & \text{if } x_r \le x < x_r + \epsilon \\ \{x - \epsilon, y\} & \text{if } x \ge x_r + \epsilon, \end{cases} \tag{11}$$

where "$\emptyset$" means that the position is eliminated. A rift either transforms a uniform hard-disk sample into a uniform sample in the reduced box, or else eliminates it because a disk falls inside the rift or because two disks overlap (see Fig. 6). In a non-periodic box, wall rifts at $x_r = 0$ or $x_r = L_x - \epsilon$ (and likewise for y) chip off a slice from the surface. Vertical rifts, as in Eq. (11), estimate the pressure P_x, and horizontal rifts ($[0, L_x] \times [y_r, y_r + \epsilon]$) estimate the pressure P_y. Simultaneous vertical and horizontal rifts with $L_y\epsilon_x = L_x\epsilon_y$ conserve the aspect ratio of the box. Equivalent to a homogeneous (homothetic) rescaling of the box, they estimate the pressure $P = (P_x + P_y)/2$.

The pressure can be estimated for finite ϵ from a finite number of samples, but this then requires an extrapolation toward $\epsilon \to 0$. In EDMD and ECMC, the extrapolation can be avoided because of the infinite number of samples produced in a given run-time interval τ_{sim}.

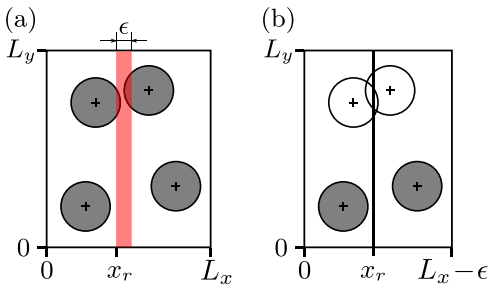

FIG. 6. Vertical rift $[x_r, x_r+\epsilon]\times[0,L_y]$. (a) $L_x\times L_y$ box with a vertical rift of width ϵ at position x_r. (b) Transformed sample, which is eliminated because of a pair overlap.

We first reduce the volume V of a non-periodic $L_x\times L_y$ box by a vertical wall rift with $x_r = L_x-\epsilon$ or by a horizontal wall rift with $y_r = L_y-\epsilon$. A sample in the original box is eliminated through the wall rift with the probability that one of the disks overlaps the wall rift. Disk i is at position $\mathbf{x}_i$ with the normalized single-disk probability density $\pi^{(1)}(\mathbf{x}_i)$ and at an x-position in the interval $[L_x-\sigma-\epsilon, L_x-\sigma]$ with probability $\epsilon\int dy\pi^{(1)}(L_x-\sigma,y)$. We normalize the single-disk density ρ^* to V, so that the normalized probability density $\pi^{(1)}(\mathbf{x}_i)$ of a given disk i to be at position $\mathbf{x}_i$ equals $\pi^{(1)}(\mathbf{x}_i) = \rho^*(\mathbf{x}_i)/V$. We then use the rescaled line density $\rho_x^*(x) = \int dy\rho^*(x,y)/L_y$, and likewise for $\rho_y^*(y)$. With the vertical rift volume ϵL_y and analogously for the horizontal one, this gives the wall-rift pressure estimator,

$$\beta\begin{bmatrix}P_x\\P_y\end{bmatrix} = \frac{N}{V}\begin{bmatrix}\rho_x^*(L_x-\sigma)\\ \rho_y^*(L_y-\sigma)\end{bmatrix}. \tag{12}$$

Naively, the rescaled line densities ρ_x^* and ρ_y^* are obtained from a histogram of the x-coordinates, extrapolated to $x = L_x-\sigma$ and equivalently to $x=\sigma$ (see Table I, line 1 and Appendix A).

Within EDMD, the rescaled line densities of Eq. (12) can be computed, without extrapolation, from the time interval $\Delta t = 2\epsilon/|v_{\text{wall}}^{\perp}|$ before and after the collision during which a disk with perpendicular velocity $v_{\text{wall}}^{\perp}$ overlaps with the wall rift at $x_r = L_x-\epsilon$. The time interval Δt here simply indexes equilibrium samples and has no kinematic meaning. The change in volume (by the two rifts at $x=0$ and $x=L_x$) equals $2\epsilon L_y$. For $\epsilon\to 0$, only a single disk overlaps with the wall rift, leading to the EDMD wall-rift estimator,

$$\beta P_x = \frac{1}{2L_y\tau_{\text{sim}}}\sum_{w:(i,\pm\hat{\mathbf{e}}_x)}\frac{2}{|v_{(i)}^{\perp}|} \tag{13a}$$

$$= \left\langle\frac{2}{|v_{\text{wall}}^{\perp}|}\right\rangle\overbrace{\frac{1}{2L_y\tau_{\text{sim}}}\sum_{w:(i,\pm\hat{\mathbf{e}}_x)}1}^{\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x}} \tag{13b}$$

$$= \frac{2\sqrt{\pi}}{\sqrt{\sum v_i^2}}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)}\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x} \tag{13c}$$

$$\xrightarrow{N\to\infty}\sqrt{2\pi\beta m}\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x}. \tag{13d}$$

The sum in Eq. (13a) goes over the wall collisions w of all disks i in the $\pm\hat{\mathbf{e}}_x$ direction, and $\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x}$ is the wall-collision rate per vertical unit line element. In Eq. (13), the right-hand sides are estimators, whose expectation yields the pressure on the left-hand side. In this equation and the following ones, the additional $\langle\ldots\rangle$ [such as $\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x}\to\langle\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x}\rangle$ in Eqs. (13c) and (13d)] are omitted. In Eq. (13a), the estimator has infinite variance. It is regularized through its mean value in Eq. (13b). The latter is evaluated in Eq. (13c) with the Maxwell-boundary expression of Eq. (8) and tested to 4×10^{-6} in relative precision against other estimators (see Table I, line 2). The pressure estimator of Eq. (13c) can also be derived as a kinematic pressure estimator through the momentum transfer with the walls (see Subsection III C 3). Thermodynamic and kinematic pressures thus agree already at finite N.

The wall-rift pressure estimator adapts non-trivially to ECMC. We consider straight ECMC with a single active disk that moves with unit speed along the $\pm\hat{\mathbf{e}}_x$ direction. As a lifted Markov chain, ECMC splits the equilibrium probability of each "collapsed" sample $\mathbf{x}$ equally between the N lifted copies (consisting of $\mathbf{x}$ and of the label of the active disk for a given displacement vector) (see Refs. 59 and 65, Appendix A). ECMC only determines overlaps with the walls for the active disk, and a lifted sample that must be eliminated is detected with a biased probability $1/N$. This bias is corrected by multiplying the right-hand side of Eq. (13a) by N, resulting in the ECMC wall-rift estimator,

$$\beta P_x = \frac{N}{2L_y\tau_{\text{sim}}}\sum_{w:(a,\pm\hat{\mathbf{e}}_x)}\frac{2}{|v_{\text{wall}}^{\perp}|} = 2N\hat{n}_{\text{wall}}^{\pm\hat{\mathbf{e}}_x} \tag{14}$$

[see Fig. 7(a)]. It is tested to 4×10^{-5} in relative precision (see Table I, line 3).

Within molecular dynamics and ECMC, pressures can also be estimated by rifts inside the box and, in particular, by averages over all rift positions x_r in addition to the average over τ_{sim} already contained in the wall-rift estimators. This can be written as

$$\beta P_x = \frac{1}{\epsilon L_y}\frac{1}{L_x\tau_{\text{sim}}}\int_t^{t+\tau_{\text{sim}}}d\tau\int_0^{L_x-\epsilon}dx_r\Theta(\tau,x_r), \tag{15}$$

TABLE I. Thermodynamic pressure estimates for four disks of radius $\sigma = 0.15$ in a non-periodic square box of sides 1. The kinematic estimators of Subsection III C 3 lead to identical expressions.

Line #	Sampling method: pressure estimator	βP
1	EDMD: wall-rift fit [see Eq. (12)]	10.74(7)
2	EDMD: wall rift [see Eq. (13c)]	10.796 25(4)
3	ECMC: wall rift [see Eq. (14)]	10.7962(4)
4	EDMD: rift average [see Eq. (19a)]	10.796 29(3)
5	ECMC: rift average [see Eq. (20)]	10.7962(4)
6	EDMD: homothetic fit [see Eq. (27a)]	10.74(4)

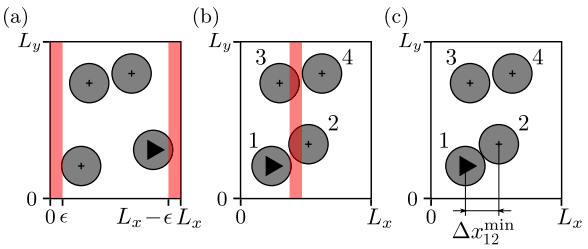

FIG. 7. ECMC rift estimators. (a) The ECMC wall-rift estimator only detects rift overlaps of the active disk, explaining the factor N in Eq. (14), which is absent in Eq. (13a). (b) A pair of disks (i,j) leading to the elimination of the sample is detected only if either i or j is active, explaining a factor $N/2$ entering Eq. (20). (c) Illustration of the x-separation at contact $\Delta x_{ij}^{\min}$ (also relevant for EDMD).

where $\Theta(\tau, x_r)$ is zero if the sample at time $t+\tau$ is maintained after the reduction with parameter x_r and one if it is eliminated. The ideal-gas contribution to Eq. (15),

$$\beta P_x^{\text{ideal gas}} = \frac{\epsilon N}{L_x}\frac{1}{\epsilon L_y} = \frac{N}{V}, \tag{16}$$

counts the proportion of rifts that eliminate samples because the centers of the disks fall inside. The pair-collision contribution to Eq. (15) is derived considering the sample shown in Fig. 6. If the distance between two disks i and j is in the interval $[2\sigma, 2\sigma + \epsilon\Delta x_{ij}^{\min}/(2\sigma)]$, where $\Delta x_{ij}^{\min}$ is the x-separation at contact, the corresponding samples are eliminated for a time $(2/|\Delta v_{ij}^{\perp}|)[\epsilon\Delta x_{ij}^{\min}/(2\sigma)]$ for vertical rifts in the interval of length $\Delta x_{ij}^{\min}$ between the two disks at contact. Together with the wall term analogous to Eq. (13), the rift-average pressure estimator for EDMD thus reads

$$\beta P_x = \frac{N}{V} + \frac{1}{V\tau_{\text{sim}}}\left[\sum_{p:(ij)} \frac{|\Delta x_{ij}^{\min}|^2}{2\sigma}\left\langle\frac{2}{\Delta v_{ij}^{\perp}}\right\rangle + \sum_{w:(i,\pm\hat{\mathbf{e}}_x)}\left\langle\frac{2\sigma}{|v_{\text{wall}}^{\perp}|}\right\rangle\right], \tag{17}$$

where the mean values again involve Maxwell-boundary expressions. Equation (17) can be combined with an analogous expression for P_y to obtain the EDMD rift-average estimator for P,

$$\beta P = \frac{N}{V} + \frac{\sigma}{V\tau_{\text{sim}}}\left[\sum_{p:(ij)}\left\langle\frac{2}{\Delta v_{ij}^{\perp}}\right\rangle + \sum_{w:(i,\pm\hat{\mathbf{e}}_x,\pm\hat{\mathbf{e}}_y)}\left\langle\frac{1}{|v_{\text{wall}}^{\perp}|}\right\rangle\right]. \tag{18}$$

In a non-periodic box, using Eqs. (8) and (9), the EDMD rift-average estimator takes the form

$$\beta P = \frac{N}{V} + \frac{\sigma\sqrt{\pi}}{\sqrt{\sum v_i^2}\,V}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)}\left(n_{\text{wall}}^{\pm\hat{\mathbf{e}}_x\pm\hat{\mathbf{e}}_y} + \sqrt{2}n_{\text{pair}}\right) \tag{19a}$$

$$\xrightarrow{N\to\infty} \frac{N}{V}\left(1 + \frac{\sigma\sqrt{\pi m\beta}}{N}n_{\text{pair}}\right), \tag{19b}$$

where $n_{\text{wall}}^{\pm\hat{\mathbf{e}}_x\pm\hat{\mathbf{e}}_y}$ is the wall-collision rate, the number of all wall collisions per time interval, and similarly for the pair-collision rate n_{pair}.

In the $N\to\infty$ limit of Eq. (19b), wall collision plays no role. The EDMD rift-average estimator of Eq. (19a) is tested to 3×10^{-6} in relative precision (see Table I, line 4).

Rift-average pressure estimators for ECMC detect wall and pair collisions with biases [see Eq. (14)], which must again be corrected, namely, by a factor N for each wall event and by a factor $N/2$ for each pair event, the latter because a lifted sample of N disks that must be eliminated is detected only if either i or j is active [see Fig. 7(c)]. This leads to the straight-ECMC rift-average estimator,

$$\beta P_x = \frac{N}{V} + \frac{N}{V\tau_{\text{sim}}}\left(\sum_{p:(i,j)}\Delta x_{ij}^{\min} + \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} 2\sigma\right), \tag{20}$$

which again differs in the factors $\propto N$ from the corresponding formulas of EDMD. Furthermore, it averages over a bounded distribution of $\Delta x_{ij}^{\min}$, with the wall-velocity only taking the values ±1, whereas in EDMD, the corresponding continuous distributions of $1/|\Delta v_{ij}^{\perp}|$ and of $1/|v_{\text{wall}}^{\perp}|$ have infinite variance. The straight-ECMC estimator of Eq. (20) is tested to 4×10^{-5} in relative precision (see Table I, line 5).

In a periodic box, there are no wall collisions, and the direction of motion of straight ECMC is either $+\hat{\mathbf{e}}_x$ or $+\hat{\mathbf{e}}_y$. In Eq. (20), all the x-separations at contact $\Delta x_{ij}^{\min}$ and the chain length τ_{sim} (which, because of the unit velocity, corresponds to the total displacement) add up to the difference of the final position x_{final} of the last disk of the chain and the initial position x_{initial} of the chain's first disk. Here, periodic boundary conditions are accounted for so that in the absence of collision, this distance equals τ_{sim}. For an event chain in the $+\hat{\mathbf{e}}_x$ direction, Eq. (20) thus simplifies into the straight-ECMC estimator for a periodic box,[43]

$$\beta P_x = \frac{N}{V\tau_{\text{sim}}}(x_{\text{final}} - x_{\text{initial}}), \tag{21}$$

that is easy to compute and that will be used extensively in Sec. IV. There, we alternate event chains in $+\hat{\mathbf{e}}_x$, which estimate P_x, and event chains in $+\hat{\mathbf{e}}_y$, which estimate P_y. Alternating the direction of straight-ECMC chains is required for convergence towards equilibrium. The rift-average estimator generalizes to other variants of ECMC. The pressure P_y can also be estimated through event chains in $\pm\hat{\mathbf{e}}_x$ and horizontal-rift averages, leading for the straight ECMC in $+\hat{\mathbf{e}}_x$ to

$$\beta P_y = \frac{N}{V} + \frac{N}{V\tau_{\text{sim}}}\sum_{p:(ij)}\frac{|\Delta y_{ij}^{\min}|^2}{\Delta x_{ij}^{\min}}. \tag{22}$$

However, this estimator has infinite variance and is less convenient than Eq. (21).

2. Homothetic volume reductions

Besides by rifts, the volume V of an $L_x\times L_y$ box can be reduced by a homothetic transformation, where the box and all positions $\mathbf{x}_i$ are homogeneously scaled by a factor $1-\epsilon_\alpha<1$, while the disk radii σ remain unchanged. The transformation of the box corresponds to simultaneous horizontal and vertical rifts of equal rift volume, but the disk positions then transform inhomogeneously, as in Eq. (11).

A homothetic volume reduction yields the pressure $\beta P = \beta(P_x + P_y)/2$, rather than one of the components. It may be performed in two steps. In the first step [from (σ, V) to (σ', V), see Fig. 8], the box and $\mathbf{x}_i$ are unchanged, but the disks are swollen by a factor $1/(1-\epsilon_\alpha)$, possibly eliminating samples. In the second step, all lengths are rescaled by $1-\epsilon_\alpha$, so that the radii return to σ. This second step [from (σ', V) to (σ, V')] is rejection-free, and its reduction of sample-space volume, with $Z(\sigma, V') = (V'/V)^N Z(\sigma', V)$, constitutes the ideal-gas term of the pressure.

In the two-stage transition $Z(\sigma, V) \to Z(\sigma', V) \to Z(\sigma, V')$, the two-step procedure turns Eq. (10) into

$$\beta P = \lim_{V' \to V} \frac{\log[Z(\sigma, V)] - \log[Z(\sigma, V')]}{V - V'} = \frac{N}{V} + \frac{1}{V - V'} \frac{Z(\sigma, V) - Z(\sigma', V)}{Z(\sigma, V)}. \tag{23}$$

The final term again divides the elimination probability of a sample by the change in volume (see also Refs. 66–69).

The pair-elimination probability is expressed by the normalized probability density $\hat{g}(r_x, r_y)$, with $\hat{g}(r_x, r_y)\mathrm{d}r_x\mathrm{d}r_y$ the probability that a given pair distance is in $[r_x, r_x + \mathrm{d}x][r_y, r_y + \mathrm{d}y]$. With $\hat{g}(r)$ as the average of $\hat{g}(r_x, r_y)$ over the corresponding ring of radius r, the probability that a given pair distance is in the interval $[r, r + \mathrm{d}r]$ is thus $2\pi r \hat{g}(r)\mathrm{d}r$. By convention, the pair correlation function $g(r)$ is normalized to $g(r) = V\hat{g}(r)$. For our application, we have $r = 2\sigma$ and $\mathrm{d}r = 2\sigma\epsilon_\alpha$, and there are $N(N-1)/2$ pairs of disks. In addition, the absolute change in volume for $L_x \to L_x(1+\epsilon_\alpha)$ and $L_y \to L_y(1+\epsilon_\alpha)$ is $2V\epsilon_\alpha$. The pair-collision contribution to the pressure is, thus,

$$\beta P^{\mathrm{pair}} = \frac{N}{V}\frac{N-1}{V} 2\pi\sigma^2 g(2\sigma), \tag{24}$$

an expression that is correct for finite N and in periodic or non-periodic boxes. The extrapolation of $g(2\sigma)$ from a histogram is detailed in Appendix A. The range of distances to the wall that are eliminated is $[\sigma, \sigma(1+\epsilon_\alpha)]$, and the change in volume remains $2\epsilon_\alpha V$. The contribution to the pressure of the wall at $x = 0$ is then

$$\beta P^{\mathrm{wall},-\hat{\mathbf{e}}_x} = \frac{N\sigma\epsilon_\alpha}{2\epsilon_\alpha V} \int \mathrm{d}y \pi^{(1)}(\sigma, y) = \frac{N\sigma}{2VL_x}\rho_x^*(\sigma), \tag{25}$$

where we used the rescaled line densities ρ_x^* and ρ_y^*, which remain $\mathcal{O}(1)$ for $V \to \infty$ (see Subsection III C 1). Summing over the four wall terms, one arrives at

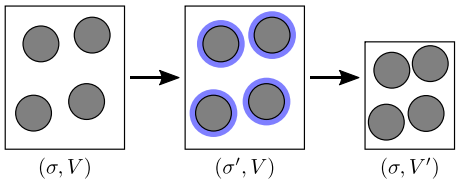

FIG. 8. A homothetic volume reduction performed through a swelling of disks followed by a uniform reduction of all lengths (box, positions, and radii).

$$\beta P^{\mathrm{wall}} = \frac{N\sigma}{V}\left[\frac{\rho_x^*(\sigma)}{L_x} + \frac{\rho_y^*(\sigma)}{L_y}\right]. \tag{26}$$

The computation of the line densities $\rho_x^*(\sigma)$ and $\rho_y^*(\sigma)$ from a histogram is detailed in Appendix A. The combined pair and wall contributions yield the homothetic pressure estimator for a non-periodic $L_x \times L_y$ box,

$$\beta P = N/V + \frac{N}{V}\left[2\pi\frac{(N-1)\sigma^2}{V} g(2\sigma) + \sigma\frac{\rho_x^*(\sigma)}{L_x} + \sigma\frac{\rho_y^*(\sigma)}{L_y}\right] \tag{27a}$$

$$\xrightarrow{N\to\infty} \frac{N}{V}[1 + 2\eta g(2\sigma)]. \tag{27b}$$

Equation (27a) is tested by histogram fits and extrapolations to contact of $\rho_x^*(\sigma)$, $\rho_y^*(\sigma)$, and $g(2\sigma)$ to 4×10^{-3} in relative precision (see Table I, line 6). Equation (27b) has long been used for estimating pressures in MCMC.[28]

EDMD and ECMC can estimate the pressure without the extrapolations of the pair correlation functions and the wall densities by tracking the time during which pairs of disks are close to contact, or a disk is close to the wall. The explicit computation for EDMD simply reproduces Eqs. (18) and (19), both at finite N and in the thermodynamic limit. The corresponding homothetic pressure estimators for ECMC are readily derived, but they have diverging variances that require specific care. For all variants except straight ECMC, they correctly estimate wall contributions to the pressure and can be used for non-periodic boxes. The velocities of straight ECMC are always parallel to some walls, precluding the estimation of all wall contribution to the pressure.

3. Kinematic pressure estimators

Kinematic pressure estimators of EDMD determine the time-averaged exchange of momentum between disks, or between disks and a wall. Their use goes back to Daniel Bernoulli,[16] who pointed out that under the scaling $\mathbf{v}_i \to \gamma\mathbf{v}_i \forall i$ of Eq. (2), both the number of collisions per time interval and the momentum transmitted scaled as γ, so that the pressure had to be proportional to the mean square velocity (in other words to the temperature). In the non-periodic $L_x \times L_y$ box, the transmitted momentum with, say, the vertical walls at $x = 0$ and $x = L_x$ gives the kinematic EDMD estimator,

$$P_x = \frac{1}{2L_y\tau_{\mathrm{sim}}} \sum_{w:(i,\pm\hat{\mathbf{e}}_x)} 2m|v_{\mathrm{wall}}^\perp| \tag{28a}$$

$$= 2m\left\langle |v_{\mathrm{wall}}^\perp| \right\rangle \hat{n}_{\mathrm{wall}}^{\pm\hat{\mathbf{e}}_x} \tag{28b}$$

$$= \frac{mR_v\sqrt{\pi}}{N}\frac{\Gamma(N+\frac{1}{2})}{\Gamma(N)} \hat{n}_{\mathrm{wall}}^{\pm\hat{\mathbf{e}}_x} \tag{28c}$$

$$\xrightarrow{N\to\infty} \sqrt{2\pi\beta m}\hat{n}_{\mathrm{wall}}^{\pm\hat{\mathbf{e}}_x}, \tag{28d}$$

where in Eq. (28c), we used Eq. (8b). Already at finite N, the kinematic EDMD estimator of Eq. (28c) is identical to the thermodynamic wall-rift pressure estimator of Eq. (13c), as we may identify $R_v^2 = 2N/(m\beta)$.

The EDMD kinetic pressure estimator can also be derived from the virial function

$$G_x = m\sum_{i=1}^{N} x_i v_{i,x}, \tag{29}$$

which is strictly bounded during molecular dynamics, so that its mean time derivative vanishes,

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}t} G_x \right\rangle = \left\langle \frac{\mathrm{d}}{\mathrm{d}t} G_x^{\mathrm{wall}} \right\rangle + \left\langle \frac{\mathrm{d}}{\mathrm{d}t} G_x^{\mathrm{pair}} \right\rangle + m\left\langle \sum_{i=1}^{N} v_{i,x}^2 \right\rangle$$
$$= m\left\langle \sum_{i=1}^{N} \left(x_i \dot{v}_{i,x} + v_{i,x}^2 \right) \right\rangle = 0. \tag{30}$$

The wall contribution to this expression stems from collisions with the vertical walls at $x = 0$ and $x = L_x$, which are given by $2m\langle|v_{\mathrm{wall}}^{\perp}|\rangle\sigma$ and $-2m\langle|v_{\mathrm{wall}}^{\perp}|\rangle(L_x - \sigma)$, respectively. This results in

$$\left\langle \frac{\mathrm{d}}{\mathrm{d}t} G_x^{\mathrm{wall}} \right\rangle = -2m\langle|v_{\mathrm{wall}}^{\perp}|\rangle(L_x/2 - \sigma) n_{\mathrm{wall}}^{\pm x} \tag{31a}$$
$$= -VP_x + 2m\sigma\langle|v_{\mathrm{wall}}^{\perp}|\rangle 2L_y \hat{n}_{\mathrm{wall}}^{\pm x}, \tag{31b}$$

where we have used Eq. (28b).

For the pair-collision contribution to Eq. (30), we use that at the collision of disks i and j, the distance $\Delta\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$ satisfies $|\Delta\mathbf{x}_{ij}| = 2\sigma$. With the unit vector $\hat{\mathbf{e}}_{\perp} = \Delta\mathbf{x}_{ij}/(2\sigma)$ and the velocity difference $\Delta\mathbf{v} = \mathbf{v}_i - \mathbf{v}_j$ (before the collision), the change in the velocity of disk i is $-\hat{\mathbf{e}}_{\perp}(\Delta\mathbf{v}\cdot\hat{\mathbf{e}}_{\perp})$ and the change in the velocity of disk j is $\hat{\mathbf{e}}_{\perp}(\Delta\mathbf{v}\cdot\hat{\mathbf{e}}_{\perp})$ (see Ref. 41, Sec. 2.1.1). An individual pair collision thus contributes

$$-m\frac{(\Delta x_{ij}^{\min})^2}{4\sigma^2}\underbrace{(\Delta\mathbf{v}\cdot\Delta\mathbf{x}_{ij}^{\min})}_{-2\sigma|\Delta v_{ij}^{\perp}|}, \tag{32}$$

an expression where both terms can be averaged independently. Finally, we may use $\langle v_{i,x}^2\rangle = R_v^2/(2N)$ to rearrange Eq. (30) into a kinematic EDMD pressure estimator,

$$\beta P_x = \frac{N}{V} + \frac{\beta}{V\tau_{\mathrm{sim}}}\sum_{w:(i,\pm\hat{\mathbf{e}}_x)}\langle 2\sigma|v_{\mathrm{wall}}^{\perp}|\rangle$$
$$+ \sum_{p:(ij)}\frac{(\Delta x_{ij}^{\min})^2}{4\sigma^2}\langle 2\sigma\Delta v_{ij}^{\perp}\rangle. \tag{33}$$

Using Eqs. (8) and (9), this kinematic estimator is seen to be equivalent to the thermodynamic rift-average estimator of Eq. (17).

IV. EQUATION-OF-STATE COMPUTATIONS

We now compare historic hard-disk pressure computations since 1953[14,28,56,70–74] with massive simulation results obtained in this work using the sampling algorithms and pressure estimators of Sec. III. Our re-evaluation will illustrate the three principal challenges that the hard-disk model shares, *mutatis mutandis*, with other sampling problems. First, the estimate of the pressure continues to depend on the initial configuration for very long run times, until equilibrium is reached. We will call this time the "mixing time"[54] in a slight abuse of terminology, as we do not consider certain pathological initial configurations, which trap the Markov chain forever (see Ref. 60). The pioneering works by Metropolis *et al.* and by Alder and Wainwright obtained crucial insights from very short computer experiments on the then available computers. Alder and Wainwright were aware of the dependence of their results on the initial configurations. They thus knew that their run times were much below the correlation-time scale. Later works that attempted to interpret manifestly unequilibrated samples,[72,73,75–77] or that failed to recognize the lack of convergence, arrived at qualitatively wrong conclusions. In the hard-disk model, mixing times can be bounded rigorously only at small densities.[78] At higher densities, heuristic criteria for the mixing time, which have not been fully presented before, appear crucial. In our case, they depend on the time series of other observables than the pressure, or on multiple runs from qualitatively different initial configurations.

The second challenge for hard-disk computations consists in the intricate dependence of the pressure on the shape of the box (that is the aspect ratio) and on the number N of disks, rendering extrapolations to the thermodynamic limit non-trivial. In small boxes, the hexatic and solid phases are confounded, as they only differ at large distances, so that the behavior in the thermodynamic limit is not necessarily reflected in the equation of state at small N.

The third challenge concerns the very evaluation of the pressure. Within Monte Carlo methods, the pressure was long evaluated through extrapolation toward contact of the pair correlation function in Eq. (27b), a procedure fraught with uncertainty. The rift formulas of Subsection III C that originated with ECMC, and that we even use in MPMC as short fictitious ECMC runs placed at regular time intervals, overcome the need for extrapolations.

A. Hard-disk equation of state for small *N*

Since the early days of computer simulation, the pressure of the hard-disk model has been computed with the aim of determining its phase behavior in the thermodynamic limit. While the identification of thermodynamic phases in finite systems can be subject to discussion, the pressure is unambiguously defined, and it can, in principle, be computed to arbitrary precision.

1. Metropolis et al. (1953), rotation criterion

In 1953, in the publication that first introduced MCMC, Metropolis *et al.*[28] estimated the pressure from the extrapolated pair-correlation function for 224 disks in a periodic square box. This number $224 = 16 \times 14$ disks can be perfectly packed at density $\eta = \pi/(2\sqrt{3}) = 0.907$ in an almost square-shaped box of aspect ratio $\alpha = (16\sqrt{3}/2 : 14) = (0.9897 : 1)$ and almost at that density in a perfectly square box. Metropolis *et al.* concluded that "(t)here is no indication of a phase transition," which, in hindsight, is explained by the historical computational limits that prevented accurate results in the transition region. The equation of state for $N = 224, \alpha = (1 : 1)$, recomputed in this work using straight ECMC to a relative precision of 10^{-4}, is somewhat higher than the historic pressures. It is also slightly non-monotonic [see Fig. 9(a)].

The 224-disk square-box system of Metropolis *et al.*, from 1953, carries lessons that are pertinent to the present day. Indeed, in a square box, any hard-disk sample can be rotated by an angle $\pi/2 = 90°$ into another valid sample. At high enough density, two

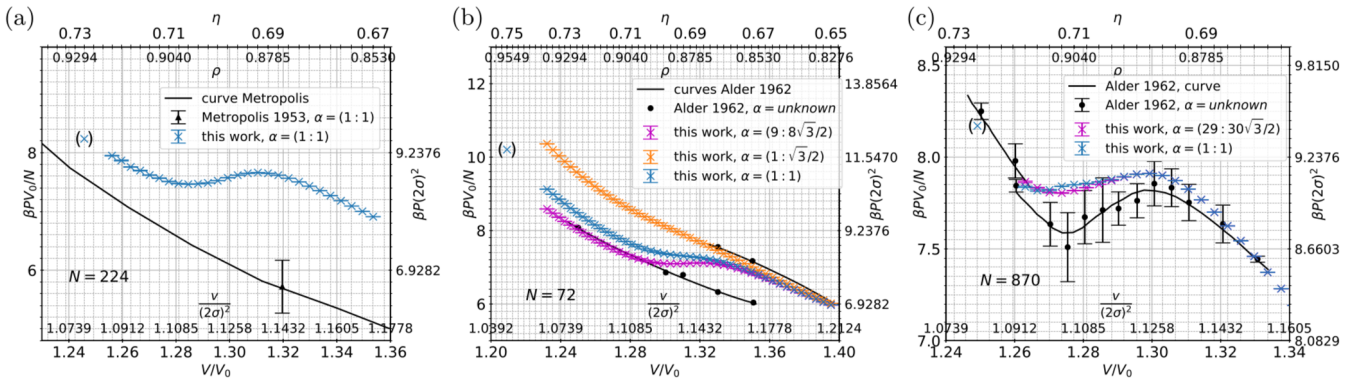

FIG. 9. Equations of state $P(V)$ for the hard-disk model at small N. (a) $P(V)$ for $N = 224$ for $\alpha = (1:1)$ computed in 1953[28] compared with ECMC computations (this work). (b) $P(V)$ for $N = 72$ computed in 1962[14] (for unspecified aspect ratio α) and ECMC pressures for $\alpha = (1:\sqrt{3}/2)$, $(1:1)$, and $(9:8\sqrt{3}/2)$. (c) $P(V)$ for $N = 870$. This work's square-box computations satisfy $\langle\Psi_6\rangle \simeq 0$, except for data points in parentheses (see Fig. 10).

such samples are inequivalent because the local hexagon, which describes the six disks that typically surround any given disk, has, on average, a 60° symmetry but not a 90° symmetry. For each local set of samples, there thus exists another inequivalent set of samples (generated through a 90° rotation) of identical statistical weight. This rotation and the corresponding transformation of samples can be formalized through the global orientational order parameter,

$$\Psi_6 = \frac{1}{N}\sum_l \frac{1}{\mathrm{nbr}(l)} \sum_{j=1}^{\mathrm{nbr}(l)} e^{6i\phi_{lj}}, \tag{34}$$

that changes from Ψ_6 to $-\Psi_6$ [that is, $\arg(\Psi_6) \to \arg(\Psi_6) + \pi$] under a rotation by 90°. In Eq. (34), ϕ_{lj} is the angle of the line connecting disks l and j with respect to the $\hat{\mathbf{e}}_x$-axis, and $\mathrm{nbr}(l)$ is the number of neighbors of l resulting from a Voronoi construction. In a square box, the ensemble average of the orientational order parameter thus satisfies $\langle\Psi_6\rangle = 0$, and for an irreducible Markov chain, it agrees with its time average, as expressed in the ergodic theorem,[54]

$$\mathbb{P}_{\pi^{\{0\}}}\left[\lim_{t\to\infty}\frac{1}{t}\sum_{i=1}^{t} f_i = \langle f\rangle\right] = 1, \tag{35}$$

where f is a function of the sample at time i given the distribution $\pi^{\{0\}}$ of initial configurations and $\mathbb{P}$ is the probability. As the mean value $\langle\Psi_6\rangle$ is known to vanish, we can employ Eq. (35) with $f = \Psi_6$ as a diagnostic tool and suppose that the hard-disk Markov chain in a square box reaches the mixing time (with errors decreasing as the square root of the run time) only when the sampled orientational order parameter Ψ_6 has covered at least one of the quadrants in the complex plane. Because of the symmetry of the box, this suffices for scanning the whole sample space. In our simulations, we confirm that Ψ_6 has been rotated more than 90° and has visited at least one of the two points on the real axis, $\Psi_6 \simeq \pm|\Psi_6|$, and one on the imaginary axis, $\Psi_6 \simeq \pm i|\Psi_6|$. In the following, we refer to this diagnostic tool as the rotation criterion. It supposes that the orientational order parameter Ψ_6 is the slowest-decaying variable in the hard-disk system. Our time series of Ψ_6 in a square box, with known mean value $\langle\Psi_6\rangle = 0$, pinpoints problems with a hard-disk pressure estimation that might not be signaled by the time series of the pressure itself. We use the rotation criterion in two different settings. In small systems (as the 224-disk case of Metropolis *et al.*), the entire range of $\arg(\Psi_6)$ values is swept through many times, leading to high-precision estimates for the pressure, even though it strongly depends on the angle. In large systems, as the hard-disk model with $N = 128^2$ at $\eta = 0.716$, which we will discuss in Fig. 12, we barely satisfy the criterion, but it still assures us that up to a symmetry $\Psi_6 \to -\Psi_6$, all relevant regions of sample space were visited. High-precision estimates for the pressure now result from the fact that the pressure depends weakly on $\arg(\Psi_6)$.

Our ECMC simulations satisfy the rotation criterion for $N = 224$ in a periodic square box up to a density of $\eta = 0.72$. At large enough densities, the ECMC simulation may remain for long times in a set of samples with essentially the same value of $\arg(\Psi_6)$ before flipping to another set of samples with $\arg(\Psi_6) + \pi$. This very slow rotation of Ψ_6 is a harbinger of the serious convergence problems of the hard-disk model for larger N at densities of physical interest. As we will show, the pressure is strongly correlated with Ψ_6 up to moderate values of N.

2. Revisiting Alder and Wainwright (1962)

Alder and Wainwright, in 1962, used EDMD to estimate the pressure for 72 and 870 disks in rectangular periodic boxes for which they did not specify the aspect ratios. As already discussed in Fig. 1, their non-monotonic equation of state led to the prediction of a phase transition. The computed pressure is independent of the sampling method (molecular dynamics, local Metropolis algorithm, ECMC), but it depends on the aspect ratio of the box. For $72 = 9 \times 8$ disks and aspect ratio $\alpha = (9 : 8\sqrt{3}/2) = (1 : 0.7698)$ where they can be perfectly packed, the equation of state obtained by ECMC in this work agrees remarkably well with the historic data [see Fig. 9(b)], an astonishing feat given the then available computers. In contrast, for a square box (aspect ratio $\alpha = (1:1)$], the equation of state follows a slight "S" shape, but it remains monotonous for all densities. For the aspect ratio $\alpha = (1 : \sqrt{3}/2)$, the pressure is barely

"S" shaped. For the aspect ratio $\alpha = (1:1)$, our ECMC computations satisfy the rotation criterion up to densities $\eta \lesssim 0.74$, and pressure estimates achieve 10^{-4} relative precision.

For 870 disks, the dependence of the pressure on the aspect ratio is less pronounced than for $N = 72$ [see Fig. 9(c)]. Since $870 = 30 \times 29$, this number of disks can be close-packed for the aspect ratio $\alpha = (29 : 30\sqrt{3}/2) = (1 : 0.896)$. For the aspect ratio $\alpha = (1:1)$, the orientation criterion is again satisfied up to high densities [see Fig. 10(a)]. However, even at moderate densities, an ECMC run can take several CPU hours before visiting all possible orientations, and the pressure clearly correlates with the orientation [see Fig. 10(b)]. On smaller time scales, the time series remains blocked in samples that all roughly have the same orientational order parameter Ψ_6 [see Fig. 10(c)]. Analyzing such a shorter time series gives incorrect estimates of the pressure (P_α or P_β, etc., rather than P). Accordingly, the window-averaged pressure features long-time deviations from the running average, with an estimated time scale of $\sim 2 \times 10^{10}$ events (corresponding to roughly two CPU hours for ECMC). Nevertheless, on a long enough time scale estimated by the rotation criterion, all these systematic errors disappear, and the error of the pressure estimator starts to decrease as the square root of the run time [see Fig. 10(d)]. The achieved 10^{-4} relative error on the pressure estimates in Fig. 9(c), from longer simulations than those illustrated in Fig. 10, is much smaller than the systematic error $|\langle P_\alpha\rangle - \langle P_\gamma\rangle|/P \sim 10^{-2}$ of a calculation that is too short to rotate Ψ_6.

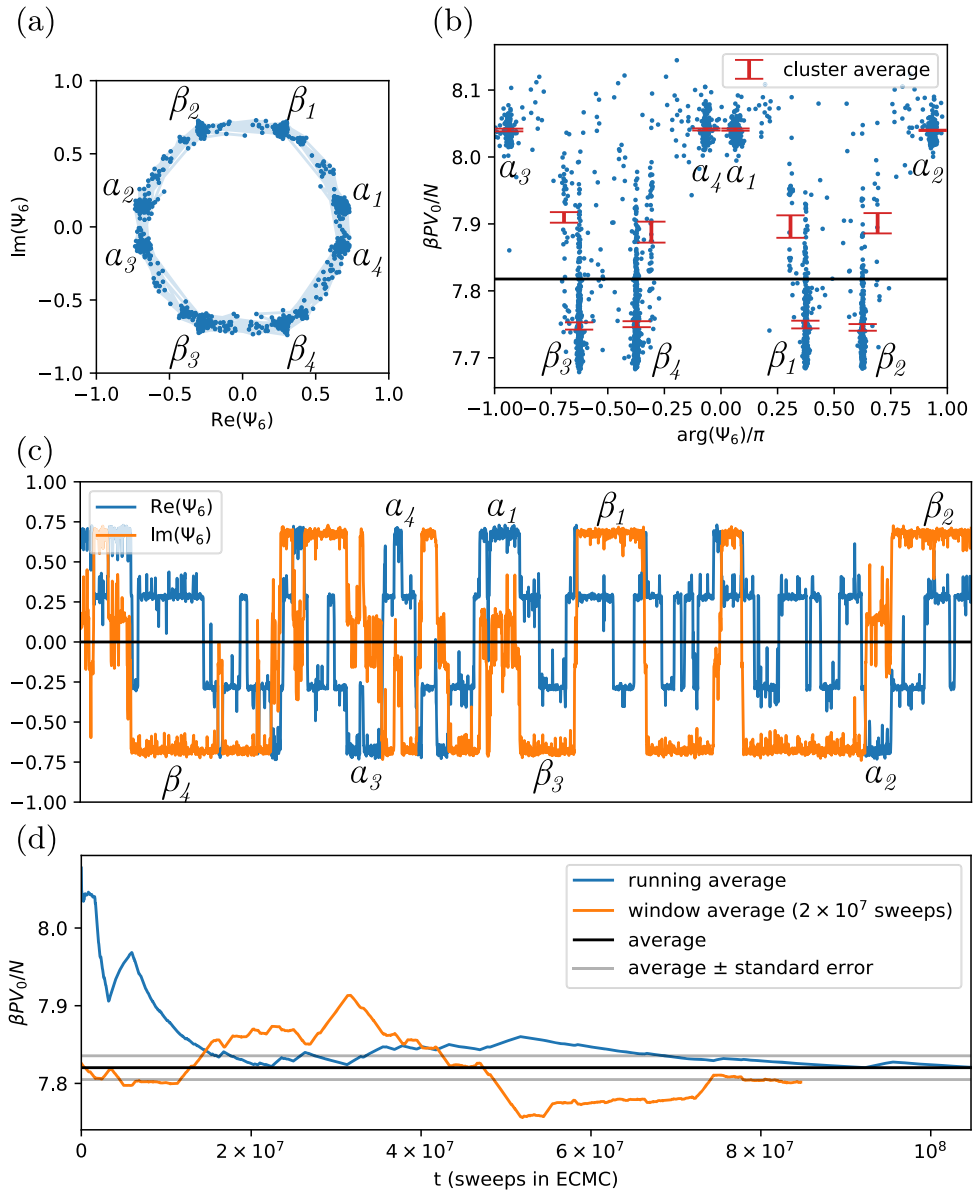

FIG. 10. Pressure P and global orientational order Ψ_6 for a three-hour ECMC run [$N = 870$, $\eta = 0.716$, square box $\alpha = (1:1)$]. (a) Values of Ψ_6 in the complex plane. Highlighted clusters with inverted Ψ_6 (such as α_1 and α_3) have the same statistical weight. (b) Cluster averages for P vs $\arg(\Psi_6)$. (c) Trajectories of $\mathrm{Im}(\Psi_6)$ and $\mathrm{Re}(\Psi_6)$ with indicated clusters. (d) Running average and window average (at the time of the beginning of the window) for P.

B. Equations of state for large *N*

The equations of state for larger N than those considered by Metropolis *et al.* and by Alder and Wainwright came into focus in the decades following 1962. At sufficiently large N, as we know today, fluid, hexatic, and solid phases can be distinguished, and the latter clearly differs from the crystal. In the simulations, three effects stand out. First, mixing and autocorrelation times become truly gigantic already for reasonable densities, even for the best currently known algorithms. Nevertheless, the pressure (as other physical quantities that we do not consider in this work) can be computed to a precision that, from a given time on, increases as the square root of the computer time. From $N = 128^2$ to $N = 1024^2$, this program can be put into practice, but it requires considerable computer resources. The failure to converge is signaled through a number of criteria, but not necessarily by the time series itself. Second, in the coexistence phase of the fluid and the hexatic in the NVT ensemble, the initial dynamics toward equilibrium is dominated by coarsening. In this process, for example, small hexatic islands nucleate in the fluid, then coalesce, and slowly grow until, in the stationary state of the time evolution, the sample presents itself as two domains, one for each coexisting phase. Precise knowledge of the pressure allows one to draw the boundaries of the phase coexistence. Third, as realized ten years ago, the high-density coexisting phase through the first-order phase transition is a hexatic and, thus, distinct from the crystal that can serve as an initial configuration of MCMC configurations. Mixing times in the hexatic phase are very long and are likely to scale with a larger exponent with N than in the fluid.[20]

1. Hard-disk model with $N = 128^2$ to $N = 512^2$

For the hard-disk model with $N = 128^2$, the relative precision levels of sequential ECMC, parallel ECMC, and MPMC reach $\sim 10^{-5}$ for the pressure, for example, at $\eta = 0.698$, much more precise than previous studies in the literature[70,79] [see Fig. 11(a)]. For $\alpha = (1:1)$, our calculations satisfy the rotation criterion up to density $\eta = 0.716$, albeit for high densities on an impressive time scale, even for the MPMC algorithm (see Fig. 12). This is where earlier studies failed to equilibrate and produced erroneous pressure estimates.

At density $\eta = 0.716$, samples of the MPMC computation may remain in one cluster indexed by a given value of $\arg(\Psi_6)$ for $\sim 10^9$ sweeps and then produce a cluster average for the pressure $\beta P V_0/N$ that differs relatively by about 10^{-3} from the equilibrium average [see Fig. 12(b)]. On such time scales, the outcome of the simulation is thus unpredictable, and the observed convergence of the pressure is not toward its ensemble average but toward some metastable cluster value. This behavior is readily detected from within the simulation data through the rotation criterion and through the dependence of obtained pressure values on initial conditions (such as different orientations or fluid and crystalline initial configurations).

For even larger systems, such as $N = 512^2$, the computations in the literature dramatically suffer from the failure to equilibrate, with incorrect pressure estimates especially at high densities. For $\alpha = (1:1)$, our MPMC implementation satisfies the rotation criterion at $\eta \lesssim 0.712$ within a few weeks of computer time (which would correspond to centuries of run time of the local Metropolis algorithm on a single CPU). For even higher densities, all currently known sampling algorithms fail to equilibrate in the strict sense of that criterion. Fortunately, at larger N, the influence of the boundary conditions is much smaller than for small N [see Fig. 13(a)]. We estimate the systematic error stemming from the failure to rotate Ψ_6 by starting independent simulations for $N = 512^2$ from a number of initial configurations with different global orientational order parameters Ψ_6 [see Fig. 13(b)]. The resulting systematic errors are found to be at most as large as the statistical errors. Our pressure values are consistent with previous ECMC and MPMC calculations[35,56] up to $\eta = 0.718$, cross-validating the correctness of the conclusion in Ref. 56 [see Fig. 11(b)]. In a non-square box, the components P_x and P_y of the pressure generally differ. For $N = 128^2$ and 512^2 at aspect ratio $\alpha = (1 : \sqrt{3}/2)$, our estimates for P_x and P_y agree within error bars even in the hexatic phase, as the system dimensions are larger than the positional correlation length.

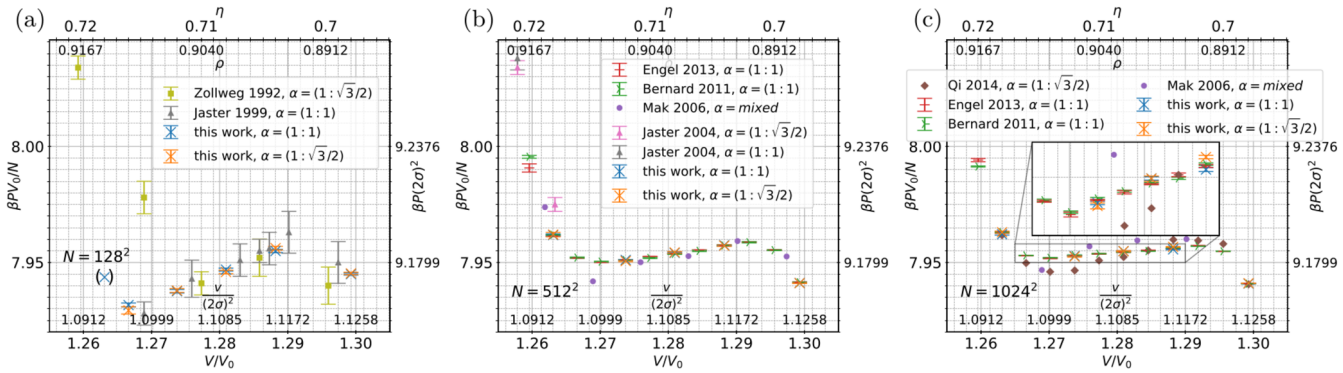

FIG. 11. Equations of state $P(V)$ for the hard-disk model at large N. (a) $P(V)$ for $N = 128^2$ from Refs. 70 and 79 and MPMC pressures (this work) for $\alpha = (1 : \sqrt{3}/2)$ and $\alpha = (1:1)$ where all but the data point in parentheses satisfy the rotation criterion (see Fig. 12 for Ψ_6-resolved pressures). (b) $P(V)$ for $N = 512^2$ from Refs. 56, 72, and 73 and MPMC pressures (this work) for aspect ratios $\alpha = (1 : \sqrt{3}/2)$ and $\alpha = (1:1)$, where runs with $\eta < 0.712$ satisfy the rotation criterion. (c) $P(V)$ for 1024^2 from Refs. 35, 56, 73, and 74, compared to MPMC (this work) for aspect ratios $\alpha = (1 : \sqrt{3}/2)$ and $(1:1)$, where at density $\eta > 0.708$, the rotation criterion is violated, but the systematic error thus committed is negligible [see Fig. 13(b)].

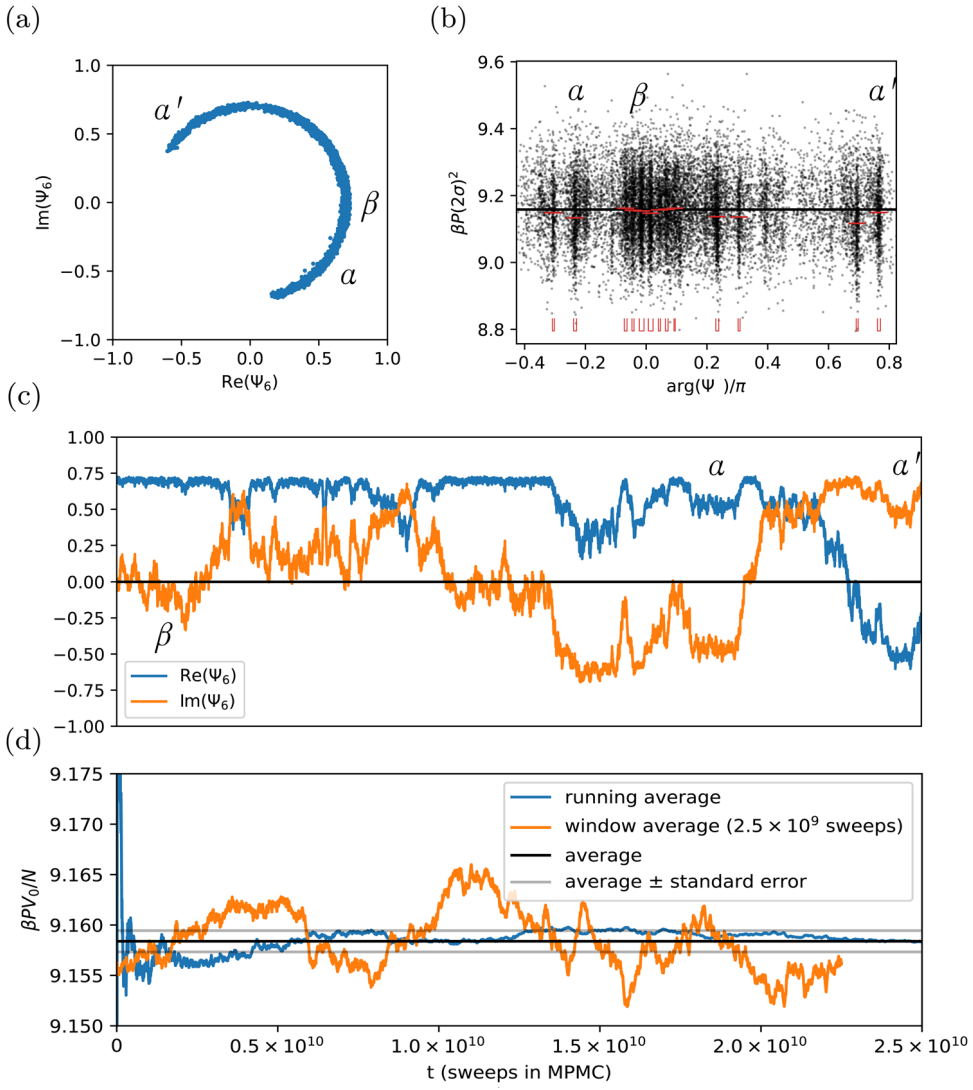

FIG. 12. Analysis of the rotation criterion for a single MPMC run [$N = 128^2$, $\eta = 0.716$, $\alpha = (1:1)$]. (a) Histogram of Ψ_6 in the complex plane. Clusters α and α' satisfy $\Psi_6^{\alpha} = -\Psi_6^{\alpha'}$ and have equal weight. (b) P vs $\arg(\Psi_6)$, showcasing the clusters. (c) Trajectory of $\mathrm{Re}(\Psi_6)$ and $\mathrm{Im}(\Psi_6)$ with first visits to clusters indicated (*cf*. Fig. 10). (d) Running average and window average (at the time of the beginning of the window) of the pressure, showcasing slow convergence.

2. Hard-disk model at $N = 1024^2$

For the hard-disk model at $N = 1024^2$, single-core implementations of the reversible Metropolis algorithm and of EDMD fail to equilibrate for densities $\eta \gtrsim 0.700$ on accessible time scales even on a modern CPU. Only straight ECMC (whose week-long mixing time of the serial version[35] reduces in the parallel implementation) and MPMC (run in parallel on thousands of cores on a GPU) are currently able to partially achieve convergence. It is for this reason that in the past, unconverged calculations of large hard-disk systems resulted in erroneous pressure estimates and, in consequence, quantitatively[74] or even qualitatively[72,73] wrong predictions for the hard-disk phases and the phase transitions.

The slow mixing manifests itself in pairs of runs that start, on the one hand, from a fluid-like initial configuration with only short-range correlations and a global orientational order parameter $|\Psi_6| \gtrsim 0$ (obtained by the Lubachevsky–Stillinger algorithm[80]) and, on the other hand, from a crystalline initial configuration with $|\Psi_6| \lesssim 1$. In the fluid–hexatic coexistence region ($\eta = 0.708$), as well as in the hexatic phase ($\eta = 0.718$), ECMC takes about 10^6 sweeps to coalesce the two values of $|\Psi_6|$ [see Figs. 13(c) and 13(d)]. For ECMC, at $\sim 10^{10}$ events/hour, this corresponds to about a week of single-core CPU time. In contrast, MPMC requires roughly 10^9 sweeps to coalesce. On a GPU with $\sim 10^4$ individual cores, this is achieved in less than two days, but on a single-core CPU, the local Metropolis algorithm (which has roughly the same efficiency per move as MPMC) would require 10^{9+6} moves, which correspond to $\sim 10^5$ h or ~ 10 years, at a typical 10^{10} moves per hour. Both branches of these calculations have similar times for arriving at equilibrium, illustrating that the fluid–hexatic coexistence phase is as difficult to reach from the fluid as it is from the crystal. While the mixing is very slow, the pairs of curves reaching the same value of $|\Psi_6|$ give a lower bound for the required run times of our ECMC and MPMC algorithms, although these times are still much below the mixing time in this system, if one were to include the rotation in $\arg(\Psi_6)$ in its

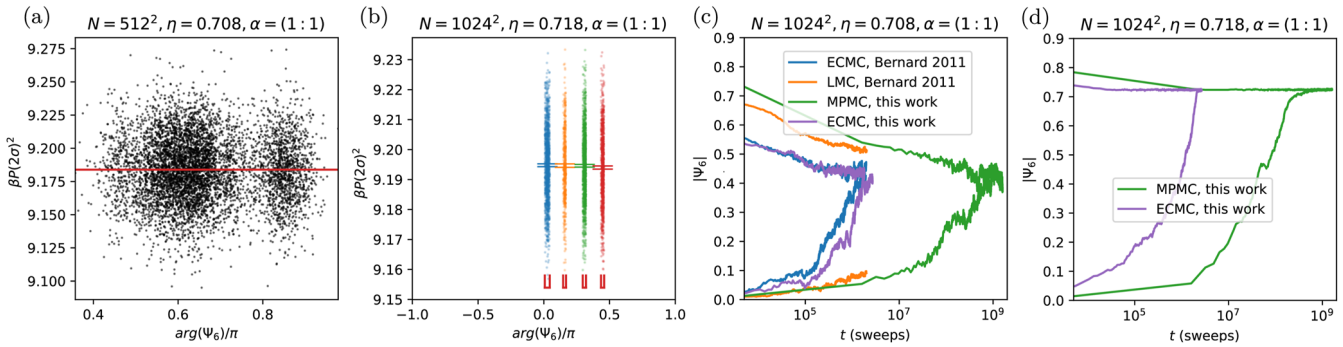

FIG. 13. Convergence analysis for the hard-disk model at $\eta = 0.708$ and $\eta = 0.718$ [square box $\alpha = (1:1)$]. (a) Scatter plot of the pressure as a function of the orientational order parameter ($N = 512^2$). (b)–(d) $N = 1024^2$ for MPMC and ECMC. (b) Cluster averages obtained from independent runs from initial configurations at specific values of Ψ_6. The difference, smaller than 10^{-3}, estimates the systematic error. (c) and (d) Time evolution of the absolute orientational order $|\Psi_6|$, starting from either a disordered initial configuration (with $|\Psi_6| \gtrsim 0$) or from a crystal (with $|\Psi_6| \lesssim 1$) (LMC refers to the local Metropolis algorithm).

definition. For the density $\eta = 0.718$, at $N = 1024^2$, our total MPMC run times amount to 6.4×10^9 sweeps, roughly 6 times longer than what is shown in Fig. 13(d).

Although MPMC and ECMC are today's fastest algorithms for the hard-disk model, they fail to satisfy the rotation criterion on human timescales for $N = 1024^2$ at densities $\eta \gtrsim 0.708$. Fortunately, the influence of $\arg(\Psi_6)$ on the pressure is quite small. To test this, we started very long MPMC calculations from a number of finely spaced crystalline initial configurations with different values of $\arg(\Psi_6)$. At the very high density of $\eta = 0.718$ for $N = 1024^2$, the relative statistical errors for the pressure is 5×10^{-4} for each run, while the maximum distance between the mean values, which could possibly correspond to a systematic error, is also found to be 5×10^{-4}. We estimate the pressure uncertainty as the maximum of the individual statistical and the difference in mean values [see Fig. 13(b)]. The independence of the estimated pressures on the aspect ratios $\alpha = (1:1)$ and $\alpha = (1:\sqrt{3}/2)$ points into the same direction [see Fig. 11(c)]. Finally, in non-square boxes, the estimates for P_x and P_y agree to very high precision for large N, while they differ markedly in smaller systems (see Table II). The disagreement of previous calculations appears thus rooted in the very long times to reach the correct values of $|\Psi_6|$.

TABLE II. Cross validation of pressure estimates between straight ECMC (naive and state-of-the-art) and MPMC in periodic boxes of given aspect ratio α, all at density $\eta = 0.698$. MPMC integrates short fictitious runs of straight ECMC, in order to estimate pressures through Eq. (21). ECMC uses the rift-average estimator of Eq. (21), except where indicated to test agreement of the pair-correlation formula of Eq. (27b).

N	α	$P/P_x, P_y$	Method
64	$(1:\sqrt{3}/2)$	8.065(3), 8.137(4)	Naive
		8.0671(9), 8.1402(9)	ECMC
72	$(9:4\sqrt{3})$	8.1614(4), 8.2382(5)	Naive
		8.1617(7), 8.2386(8)	ECMC
256^2	$(1:1)$	9.172(5)	ECMC $[g(2\sigma)]$
		9.1707(2)	ECMC
		9.1708(1)	MPMC
256^2	$(1:\sqrt{3}/2)$	9.176(6)	ECMC $[g(2\sigma)]$
		9.1703(2), 9.1704(3)	ECMC
		9.1704(1), 9.1705(1)	MPMC
512^2	$(1:1)$	9.170(2)	ECMC $[g(2\sigma)]$
		9.1699(2)	ECMC
		9.1696(1)	MPMC
512^2	$(1:\sqrt{3}/2)$	9.167(3)	ECMC $[g(2\sigma)]$
		9.1694(3), 9.1694(3)	ECMC
		9.1695(2), 9.1697(2)	MPMC

V. CONCLUSION

In this work, we have discussed the hard-disk pressure, which was estimated in the very first MCMC computation in 1953, and in one of the earliest molecular-dynamics computations, in 1962. We have argued that the difficulty of the pressure estimation had not been fully realized in the decades-long controversy over the phase-transition scenario of this simple model. Our first aim was to provide the context for this computation through a discussion of the physics of the hard-disk model, of the sampling algorithms and pressure estimators and, crucially, of the criteria for bounding mixing times. Our second aim was to finally provide definite high-precision estimates of the pressure through massive computations and to compare them to the values from the literature, thereby ending a long period of uncertainty and doubt. In doing so, we hope to provide benchmarks for the next generation of sampling algorithms, estimators, and physical theories.

The history of the hard-disk model epitomizes a number of prime computational issues. One of them is the role of so-called "computer experiments," that is, of heuristic simulations that run for much less than the mixing time. The pioneering work of Alder and Wainwright was clearly of that type, as their published pressures explicitly depend on the initial configurations.

Computer experiments below the mixing-time scale are akin to perturbation expansions in the theory of liquids or in quantum

mechanics, as the sampling below the mixing-time scale merely "perturbs" around the crystalline or fluid initial configurations. Just like perturbation theory, such computer experiments can provide important insights, yet they have limited predictive power, as was evidenced by the decades-long controversy about the hard-disk phase transitions. Beyond the mixing time, the influence of the initial configuration fades away exponentially, and exponential convergence toward the equilibrium distribution sets in. Only the statistical errors remain. In this regime, MCMC and molecular-dynamics sampling unfolds all its power. Although the mixing and correlation time scales can be gigantic, as discussed, the goal of sampling beyond the mixing time scale must not be lost sight of.

A crucial computational issue for MCMC and molecular-dynamics algorithms consists of estimating the mixing times. We have insisted that the analysis of a single time series [as that of the pressure in Fig. 12(d)] may have to be supplemented by additional cues. In this sense, we have discussed two strategies to estimate these times reliably for the hard-disk model. First, we designed an observable—the orientational order parameter for hard disks in a square box—with a known mean value. We then argued that as long as the run-time average of this observable differed considerably from its known mean, the mixing-time scale has not been reached. Used for more than a decade,[35,36] this rotation criterion supposes that the orientational order is the slowest-moving observable in the hard-disk system.

Our second strategy to lend credibility to our MCMC calculations consists of starting from widely different initial configurations, following in the footsteps of Alder and Wainwright, yet accepting the result of the calculation only if the influence of the initial configuration has faded away. This approach is related to the coupling approach for Markov chains.[81] The fluid and crystalline initial configurations that we used to initialize Markov chains for 10^6 disks in the hexatic phase stand in for the worst-case initializations, as they are called for in the definition of the mixing time.[54] The mixing time provides the relevant time scale for analyzing MCMC calculations, and certainly the one where run-time averages become independent of how the Markov chain is initialized.

Finally, we emphasize the role of algorithm development, and of hardware implementations, even in the simple model of hard disks. In this work, we relied heavily on ECMC, which, as evidenced in Figs. 13(c) and 13(d), speeds up MCMC simulations by several orders of magnitude. ECMC is a family of non-reversible Markov chains, rather than a specific algorithm, and variants of the original straight and reflective ECMC continue to be developed. The opportunities granted by non-reversible Markov chains (and by MCMC approaches in general), are certainly very far from having all been explored. The recent extension of ECMC to arbitrary interaction potentials[43] and, in particular, to the field of molecular simulation[82,83] carries considerable promise. The spectacular development of GPU hardware over the last fifteen years has greatly democratized parallel computations with, again, one of the cleanest applications being the hard-disk model. Decidedly, this simple model is a "*Drosophila*" of statistical physics.

ACKNOWLEDGMENTS

P.H. acknowledges support from the Studienstiftung des deutschen Volkes and from Institut Philippe Meyer. W.K. acknowledges support from the Alexander von Humboldt Foundation. The authors thank R.E. Kohler for helpful correspondence. Figure 1 is adapted with permission from Ref. 14. Copyrighted by the American Physical Society.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

Botao Li: Conceptualization (supporting); Data curation (equal); Investigation (equal); Software (equal); Validation (equal); Visualization (lead); Writing – original draft (supporting); Writing – review & editing (supporting). **Yoshihiko Nishikawa**: Conceptualization (equal); Data curation (equal); Investigation (equal); Software (lead); Validation (equal); Visualization (supporting); Writing – original draft (supporting); Writing – review & editing (supporting). **Philipp Höllmer**: Data curation (supporting); Software (equal); Validation (equal); Writing – review & editing (supporting). **Louis Carillo**: Conceptualization (supporting); Data curation (equal); Investigation (supporting); Software (supporting); Visualization (supporting). **A. C. Maggs**: Conceptualization (equal); Investigation (equal); Software (supporting); Validation (equal); Writing – original draft (equal); Writing – review & editing (equal). **Werner Krauth**: Conceptualization (lead); Investigation (lead); Software (supporting); Supervision (lead); Validation (equal); Writing – original draft (lead); Writing – review & editing (lead).

DATA AVAILABILITY

All materials (data and algorithms) are available through the open-source software project.

APPENDIX A: EXTRAPOLATION AND STATISTICS

Sampling algorithms output time series of configurations and of pressures (for example, one value of the estimated P_x for each event chain in $\pm\hat{\mathbf{e}}_x$). Further analysis transforms this raw output into the pressure estimates and confidence intervals provided with this work. The pressure estimators that rely on the extrapolation of pair-correlation functions and wall densities have been superseded in recent years by the rift-average estimators. We, nevertheless, describe them here in order to illustrate that the new estimators are perfectly sound. We also sketch the stationary-bootstrap method, which estimates the confidence intervals of the pressure time series.

1. Extrapolation of pair correlations and wall densities

The pressure estimator of Eq. (27a) extrapolates the rescaled line densities $\rho_x^*(x)$ and $\rho_y^*(y)$ to $x = \sigma, L_x - \sigma$ and $y = \sigma, L_y - \sigma$, respectively, and the pair-correlation function $g(r)$ to contact at $r = 2\sigma$. We use the fourth-order polynomial histogram fitting procedure of Ref. 35 contained in the `HistoricDisks` software package (see Appendix B). Within our MPMC production runs, however, we use the parameter-free rift-average estimators from fictitious straight-ECMC runs to estimate P_x and P_y, rather than the extrapolation method.

To determine the rescaled line density $\rho_x^*(\sigma)$ (and similarly ρ_y^*), the x-coordinate of a disk at position $\mathbf{x}_i = (x_i, y_i)$ is retained in a histogram of bin size $10^{-3}\sigma$ if $x_i < 1.1\sigma$ or $x_i > L_x - 1.1\sigma$. The histogram is normalized by dividing the number of elements in each bin by $2 \times 10^{-3}\sigma nN$, where n is the total number of sampled configurations (not only those contributing to the histogram) and N is the number of disks. The histogram is further multiplied by L_x for ρ_x^* (and likewise by L_y for ρ_y^*) in order to satisfy the normalization $\pi^{(1)}(\mathbf{x}_i) = \rho^*(\mathbf{x}_i)/V$. It is the line density $\rho_x^*(x)$, which is fitted and then extrapolated to $x = \sigma$.

The extrapolation of $g(2\sigma)$ proceeds analogously to that of $\rho_x^*(x)$. The pair distances in the range $2\sigma < r < 2.1\sigma$ are retained in a histogram, then normalized by dividing the number of elements in each bin by the bin size $10^{-3}\sigma$ and the total number of sampled distances $nN(N-1)/2$. The normalized histogram approximates $2\pi r\hat{g}(r)$. The histogram is further multiplied by $V/2\pi$ and divided by the distance r corresponding to the center of each bin, yielding the empirical $g(r)$, that is then extrapolated to $r = 2\sigma$.

2. Statistics

The standard errors in this work were computed using the stationary bootstrap method[84,85] and double-checked using the blocking method.[86] In stationary bootstrap, the standard error is estimated by creating a large number of simulated time series (typically 1000). Each of the time series has the same length as the original series and is created by piecing together randomly chosen sub-series of geometrically distributed length. The only parameter controlling the geometrical distribution is chosen so that it minimizes the mean squared error of the estimated standard error for an infinite sub-series length and for an infinite number of sub-series.[87,88] The compatibility of the stationary-bootstrap error estimate with that of the blocking method was carefully checked for the entire data presented in the figures as well as in Table II.

APPENDIX B: HISTORIC DATA, CODES, AND VALIDATION

The present work is accompanied by the `HistoricDisks` data and software package, which is published as an open-source project under the GNU GPLv3 license. `HistoricDisks` is available on GitHub as a part of the JeLLyFysh organization.[89] The package provides the pressure data extracted from the literature since 1953, and also the set of high-precision pressures of the present work (see Subsection B 1). Furthermore, the package contains naive MCMC and MD implementations and pressure estimators (used for validation purposes in Table I) as well as state-of-the-art implementations used in Sec. IV.

1. Pressure data, equations of states

The pressure data in the `HistoricDisks` package are from Refs. 14, 28, 35, 70, 72–74, and 79, or else correspond to results obtained in this work. Pressure data for a given reference, a given system size and aspect ratio are stored in a separate file in the .csv format (see the README file for details). Pressures and error bars were digitized using the WebPlotDigitizer software[90] where applicable, or else extracted from published tables. The `HistoricDisks` package furthermore provides Python programs that visualize equations of state. All pressure data are for the NVT ensemble, and the control variable (volume or density, plotted on the x-axis) follows all four conventions of Eq. (1). The dependent variable (the pressure, plotted on the y-axis) follows two conventions, namely, $\beta P V_0/N$ and $\beta P(2\sigma)^2$. In order to facilitate the direct comparison across different conventions, the produced figures have four x-axes and two y-axes. The pressure database in the `HistoricDisks` package may evolve in the future.

2. Computer programs

In addition to pressure data, the `HistoricDisks` package provides access to sampling algorithms (the local Metropolis algorithm, EDMD, and several variants of ECMC). Each algorithm is implemented in two versions. A naive version for four disks in a non-periodic rectangular box is patterned after Ref. 41. A naive version for N disks in a periodic rectangular box is useful for validation of more advanced methods. Both versions are implemented in Python3 (compatible with PyPy3). In addition, the package provides a state-of-the-art ECMC program for hard disks.

a. Four-disk non-periodic-box programs

Our naive programs consider four disks of radius $\sigma = 0.15$ in a non-periodic square box of sides 1. We implement the Metropolis algorithm, EDMD, and the straight, reflective,[36] forward[61] and Newtonian[62] variants of ECMC. In addition, the pressure estimators discussed in Subsection III C are implemented (see Table I). In detail, we provide pressure estimates from the wall density (using fit of the histogram), from the wall rifts using EDMD and the wall rifts using ECMC, the latter testing the bias factor N that is introduced because ECMC only moves a single disk. Moreover, we check our rift-average estimators for EDMD and for straight ECMC (that again differ by different biasing factors and mean values of perpendicular velocity components). Finally, we provide a test of the traditional fitting formula involving the pair-correlation function. All these estimators are of thermodynamic origin. As discussed in the main text, the kinematic estimators, including the virial formula, lead to identical formulas and need not be tested independently.

b. Naive periodic-box programs

The naive periodic-box programs contained in the `HistoricDisks` package differ from the naive programs only in that the number N of disks and the radius σ can be set freely and that the box is periodic. These programs have some use for demonstration purposes and to test the more efficient algorithms for relatively small values of N. Again, the Metropolis algorithm, EDMD, and the four variants of ECMC are implemented. Runs start from a crystalline initial configuration. Configurations are output at fixed time intervals. Straight ECMC also outputs estimates of the pressure.

c. State-of-the-art hard-disk programs

The `HistoricDisks` package contains an optimized C++ code for straight ECMC, which is derived from the Fortran90 code used in Ref. 35. The GPU-based MPMC Cuda code used in this work derives from a general MPMC code for soft-sphere models and will be published elsewhere.[91] Pressures obtained from these implementations agree within very tight error bars (see Table II). A Python script contained in the package analyzes samples that were saved from these

two codes in the HDF5[92] file format. It computes, for example, the global orientational order parameters Ψ_6.

REFERENCES

[1] P. Kapitza, "Viscosity of liquid helium below the λ-point," Nature **141**, 74 (1938).
[2] V. G. Vaks and A. I. Larkin, "On phase transitions of second order," J. Exp. Theor. Phys. **22**, 678 (1966), available at http://www.jetp.ras.ru/cgi-bin/e/index/e/22/3/p678?a=list.
[3] M. Campostrini, M. Hasenbusch, A. Pelissetto, and E. Vicari, "Theoretical estimates of the critical exponents of the superfluid transition in ^{4}He by lattice methods," Phys. Rev. B **74**, 144506 (2006).
[4] M. Hasenbusch, "Monte Carlo study of an improved clock model in three dimensions," Phys. Rev. B **100**, 224517 (2019).
[5] S. M. Chester, W. Landry, J. Liu, D. Poland, D. Simmons-Duffin, N. Su, and A. Vichi, "Carving out OPE space and precise O(2) model critical exponents," J. High Energy Phys. **2020**, 142.
[6] E. L. Pollock and D. M. Ceperley, "Path-integral computation of superfluid densities," Phys. Rev. B **36**, 8343 (1987).
[7] K. G. Wilson, "The renormalization group and critical phenomena," Rev. Mod. Phys. **55**, 583 (1983).
[8] J. A. Lipa, D. R. Swanson, J. A. Nissen, T. C. P. Chui, and U. E. Israelsson, "Heat capacity and thermal relaxation of bulk helium very near the lambda point," Phys. Rev. Lett. **76**, 944 (1996).
[9] J. A. Lipa, J. A. Nissen, D. A. Stricker, D. R. Swanson, and T. C. P. Chui, "Specific heat of liquid helium in zero gravity very near the lambda point," Phys. Rev. B **68**, 174518 (2003).
[10] H. J. Muller, "Nobel Lecture: The production of mutations," in *Nobel Lectures, Physiology or Medicine 1942-1962* (Elsevier Publishing Company, Amsterdam, 1964), available at https://www.nobelprize.org/prizes/medicine/1946/muller/lecture/.
[11] C. Nüsslein-Volhard, "Nobel Lecture: The identification of genes controlling development in flies and fishes," in *Nobel Lectures, Physiology or Medicine 1991-1995*, edited by N. Ringertz (World Scientific Publishing Co., Singapore, 1997), available at https://www.nobelprize.org/prizes/medicine/1995/nusslein-volhard/lecture/.
[12] R. E. Kohler, *Lords of the Fly: Drosophila Genetics and the Experimental Life, History, Philosophy, and Social Studies of Science: Biology* (University of Chicago Press, 1994).
[13] M. R. Dietrich, R. A. Ankeny, and P. M. Chen, Publ. Trends Model Org. Res., Genet. **198**, 787 (2014).
[14] B. J. Alder and T. E. Wainwright, "Phase transition in elastic disks," Phys. Rev. **127**, 359 (1962).
[15] J. M. Kosterlitz and D. J. Thouless, "Ordering, metastability and phase transitions in two-dimensional systems," J. Phys. C: Solid State Phys. **6**, 1181 (1973).
[16] D. Bernoulli, *Hydrodynamica* (ETH-Bibliothek Zürich, 1738); available at https://doi.org/10.3931/e-rara-3911.
[17] Y. G. Sinai, "Dynamical systems with elastic reflections," Russ. Math. Surv. **25**, 137 (1970).
[18] N. Simányi, "Proof of the Boltzmann-Sinai ergodic hypothesis for typical hard disk systems," Invent. Math. **154**, 123 (2003).
[19] J. L. Lebowitz and O. Penrose, "Convergence of virial expansions," J. Math. Phys. **5**, 841 (1964).
[20] T. Helmuth, W. Perkins, and S. Petti, "Correlation decay for hard spheres via Markov chains," Ann. Appl. Probab. **32**, 2063 (2022).
[21] L. Boltzmann, *Lectures on Gas Theory, Dover Books on Physics* (Dover Publications, 1995).
[22] L. Fejes, "Über einen geometrischen Satz," Math. Z. **46**, 83 (1940).
[23] T. Richthammer, "Translation-invariance of two-dimensional Gibbsian point processes," Commun. Math. Phys. **274**, 81 (2007).
[24] T. Richthammer, "Lower bound on the mean square displacement of particles in the hard disk model," Commun. Math. Phys. **345**, 1077 (2016).
[25] J. G. Kirkwood and E. Monroe, "Statistical mechanics of fusion," J. Chem. Phys. **9**, 514 (1941).
[26] G. Battimelli and G. Ciccotti, "Berni Alder and the pioneering times of molecular simulation," Eur. Phys. J. H **43**, 303 (2018).
[27] R. Peierls, "Quelques propriétés typiques des corps solides," Ann. Inst. Henri Poincare **5**, 177 (1935), available at https://eudml.org/doc/78996.
[28] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," J. Chem. Phys. **21**, 1087 (1953).
[29] H. E. Stanley and T. A. Kaplan, "Possibility of a phase transition for the two-dimensional Heisenberg model," Phys. Rev. Lett. **17**, 913 (1966).
[30] N. D. Mermin, "Crystalline order in two dimensions," Phys. Rev. **176**, 250 (1968).
[31] J. E. Mayer and W. W. Wood, "Interfacial tension effects in finite, periodic, two-dimensional systems," J. Chem. Phys. **42**, 4268 (1965).
[32] J. M. Kosterlitz, "The critical properties of the two-dimensional xy model," J. Phys. C: Solid State Phys. **7**, 1046 (1974).
[33] B. I. Halperin and D. R. Nelson, "Theory of two-dimensional melting," Phys. Rev. Lett. **41**, 121 (1978).
[34] A. P. Young, "Melting and the vector Coulomb gas in two dimensions," Phys. Rev. B **19**, 1855 (1979).
[35] E. P. Bernard and W. Krauth, "Two-step melting in two dimensions: First-order liquid-hexatic transition," Phys. Rev. Lett. **107**, 155704 (2011).
[36] E. P. Bernard, W. Krauth, and D. B. Wilson, "Event-chain Monte Carlo algorithms for hard-sphere systems," Phys. Rev. E **80**, 056704 (2009).
[37] J. Lee and K. J. Strandburg, "First-order melting transition of the hard-disk system," Phys. Rev. B **46**, 11190 (1992).
[38] H. C. Andersen, "Molecular dynamics simulations at constant pressure and/or temperature," J. Chem. Phys. **72**, 2384 (1980).
[39] M. Parrinello and A. Rahman, "Polymorphic transitions in single crystals: A new molecular dynamics method," J. Appl. Phys. **52**, 7182 (1981).
[40] L. A. Rowley, D. Nicholson, and N. G. Parsonage, "Monte Carlo grand canonical ensemble calculation in a gas-liquid transition region for 12-6 Argon," J. Comput. Phys. **17**, 401 (1975).
[41] W. Krauth, *Statistical Mechanics: Algorithms and Computations* (Oxford University Press, 2006).
[42] S. Asakura and F. Oosawa, "On interaction between two bodies immersed in a solution of macromolecules," J. Chem. Phys. **22**, 1255 (1954).
[43] M. Michel, S. C. Kapfer, and W. Krauth, "Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps," J. Chem. Phys. **140**, 054116 (2014).
[44] B. J. Alder and T. E. Wainwright, "Phase transition for a hard sphere system," J. Chem. Phys. **27**, 1208 (1957).
[45] B. J. Alder and T. E. Wainwright, "Studies in molecular dynamics. I. General method," J. Chem. Phys. **31**, 459 (1959).
[46] D. C. Rapaport, "The event scheduling problem in molecular dynamic simulation," J. Comput. Phys. **34**, 184 (1980).
[47] D. C. Rapaport, "The event-driven approach to N-body simulation," Prog. Theor. Exp. Phys. **178**, 5 (2009).
[48] M. Isobe, "Simple and efficient algorithm for large scale molecular dynamics simulation in hard disk system," Int. J. Mod. Phys. C **10**, 1281 (1999).
[49] M. Isobe, "Hard sphere simulation in statistical physics—Methodologies and applications," Mol. Simul. **42**, 1317 (2016).
[50] M. N. Bannerman, R. Sargant, and L. Lue, "DynamO: A free $\mathcal{O}(N)$ general event-driven molecular dynamics simulator," J. Comput. Chem. **32**, 3329 (2011).
[51] B. D. Lubachevsky, "Simulating billiards: Serially and in parallel," Int. J. Comput. Simul. **2**, 373 (1992).
[52] S. Miller and S. Luding, "Event-driven molecular dynamics in parallel," J. Comput. Phys. **193**, 306 (2004).
[53] M. A. Khan and M. C. Herbordt, "Parallel discrete molecular dynamics simulation with speculation and in-order commitment," J. Comput. Phys. **230**, 6563 (2011).
[54] D. A. Levin, Y. Peres, and E. L. Wilmer, *Markov Chains and Mixing Times* (American Mathematical Society, 2008).
[55] J. A. Anderson, E. Jankowski, T. L. Grubb, M. Engel, and S. C. Glotzer, "Massively parallel Monte Carlo for many-particle simulations on GPUs," J. Comput. Phys. **254**, 27 (2013).

[56]M. Engel, J. A. Anderson, S. C. Glotzer, M. Isobe, E. P. Bernard, and W. Krauth, "Hard-disk equation of state: First-order liquid-hexatic transition in two dimensions with three simulation methods," Phys. Rev. E **87**, 042134 (2013).
[57]F. Chen, L. Lovász, and I. Pak, "Lifting Markov chains to speed up mixing," in *Proceedings of the 17th Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, 1999), p. 275.
[58]P. Diaconis, S. Holmes, and R. M. Neal, "Analysis of a nonreversible Markov chain sampler," Ann. Appl. Probab. **10**, 726 (2000).
[59]W. Krauth, "Event-chain Monte Carlo: Foundations, applications, and prospects," Front. Phys. **9**, 229 (2021).
[60]P. Höllmer, N. Noirault, B. Li, A. C. Maggs, and W. Krauth, "Sparse hard-disk packings and local Markov chains," J. Stat. Phys. **187**, 31 (2022).
[61] M. Michel, A. Durmus, and S. Sénécal, "Forward Event-Chain Monte Carlo: Fast sampling by randomness control in irreversible Markov chains," J. Comput. Graph. Stat. **29**, 689 (2020).
[62]M. Klement and M. Engel, "Efficient equilibration of hard spheres with Newtonian event chains," J. Chem. Phys. **150**, 174108 (2019).
[63]B. Li, S. Todo, A. C. Maggs, and W. Krauth, "Multithreaded event-chain Monte Carlo with local times," Comput. Phys. Commun. **261**, 107702 (2021).
[64]B. Li, Y. Nishikawa, A. C. Maggs, and W. Krauth, "Multithreaded event-chain Monte Carlo: Implementation and benchmarks" (2022) (unpublished).
[65]L. Qin, P. Höllmer, and W. Krauth, "Direction-sweep Markov chains," J. Phys. A: Math. Theor. **55**, 105003 (2022).
[66]R. Eppenga and D. Frenkel, "Monte Carlo study of the isotropic and nematic phases of infinitely thin hard platelets," Mol. Phys. **52**, 1303 (1984).
[67]P. E. Brumby, A. J. Haslam, E. de Miguel, and G. Jackson, "Subtleties in the calculation of the pressure and pressure tensor of anisotropic particles from volume-perturbation methods and the apparent asymmetry of the compressive and expansive contributions," Mol. Phys. **109**, 169 (2011).
[68]E. de Miguel and G. Jackson, "The nature of the calculation of the pressure in molecular simulations of continuous models from volume perturbations," J. Chem. Phys. **125**, 164109 (2006).
[69]M. P. Allen, "Evaluation of pressure tensor in constant-volume simulations of hard and soft convex bodies," J. Chem. Phys. **124**, 214103 (2006).
[70]J. A. Zollweg and G. V. Chester, "Melting in two dimensions," Phys. Rev. B **46**, 11186 (1992).
[71] A. Jaster, "Computer simulations of the two-dimensional melting transition using hard disks," Phys. Rev. E **59**, 2594 (1999).
[72]A. Jaster, "The hexatic phase of the two-dimensional hard disks system," Phys. Lett. A **330**, 120 (2004).
[73]C. H. Mak, "Large-scale simulations of the two-dimensional melting of hard disks," Phys. Rev. E **73**, 065104 (2006).
[74]W. Qi, A. P. Gantapara, and M. Dijkstra, "Two-stage melting induced by dislocations and grain boundaries in monolayers of hard spheres," Soft Matter **10**, 5449 (2014).
[75]H. Weber, D. Marx, and K. Binder, "Melting transition in two dimensions: A finite-size scaling analysis of bond-orientational order in hard disks," Phys. Rev. B **51**, 14636 (1995).
[76]A. C. Mitus, H. Weber, and D. Marx, "Local structure analysis of the hard-disk fluid near melting," Phys. Rev. E **55**, 6855 (1997).
[77]K. Binder, S. Sengupta, and P. Nielaba, "The liquid-solid transition of hard discs: First-order transition or Kosterlitz-Thouless-Halperin-Nelson-Young scenario?," J. Phys.: Condens. Matter **14**, 2323 (2002).
[78]R. Kannan, M. W. Mahoney, and R. Montenegro, "Rapid mixing of several Markov chains for a hard-core model," in *Proceedings of 14th annual ISAAC, Lecture Notes in Computer Science* (Springer, Berlin, Heidelberg, 2003), pp. 663–675.
[79]A. Jaster, "An improved Metropolis algorithm for hard core systems," Physica A **264**, 134 (1999).
[80]B. D. Lubachevsky and F. H. Stillinger, "Geometric properties of random disk packings," J. Stat. Phys. **60**, 561 (1990).
[81] J. G. Propp and D. B. Wilson, "Exact sampling with coupled Markov chains and applications to statistical mechanics," Random Struct. Algorithms **9**, 223 (1996).
[82]M. F. Faulkner, L. Qin, A. C. Maggs, and W. Krauth, "All-atom computations with irreversible Markov chains," J. Chem. Phys. **149**, 064113 (2018).
[83]P. Höllmer, L. Qin, M. F. Faulkner, A. C. Maggs, and W. Krauth, "JELLYFYSH-Version1.0—A Python application for all-atom event-chain Monte Carlo," Comput. Phys. Commun. **253**, 107168 (2020).
[84]D. N. Politis and J. P. Romano, "The stationary bootstrap," J. Am. Stat. Assoc. **89**, 1303 (1994).
[85]Y. Nishikawa, J. Takahashi, and T. Takahashi, "Stationary bootstrap: a refined error estimation for equilibrium time series," arXiv:2112.11837 (2021).
[86]H. Flyvbjerg and H. G. Petersen, "Error estimates on averages of correlated data," J. Chem. Phys. **91**, 461 (1989).
[87]D. N. Politis and H. White, "Automatic block-length selection for the dependent bootstrap," Econ. Rev. **23**, 53 (2004).
[88]A. Patton, D. N. Politis, and H. White, "'Correction to' Automatic block-length selection for the dependent bootstrap" by D. Politis and H. White," Econ. Rev. **28**, 372 (2009).
[89]The url of the repository is https://github.com/jellyfysh/HistoricDisks.
[90]A. Rohatgi, Webplotdigitizer: Version 4.5, 2021.
[91] Y. Nishikawa, W. Krauth, and A. C. Maggs, "Two-dimensional soft spheres - phase diagrams and phase transitions" (2022) (unpublished).
[92]The HDF Group, Hierarchical Data Format, version 5 (1997–2022).

Publication 4: SGD with a constant large learning rate can converge to local maxima

SGD WITH A CONSTANT LARGE LEARNING RATE CAN CONVERGE TO LOCAL MAXIMA

Liu Ziyin[1], **Botao Li**[2], **James Simon**[3], **& Masahito Ueda**[1]
[1]The University of Tokyo
[2]ENS, Université PSL, CNRS, Sorbonne Université, Université de Paris
[3]University of California, Berkeley

ABSTRACT

Previous works on stochastic gradient descent (SGD) often focus on its success. In this work, we construct worst-case optimization problems illustrating that, when not in the regimes that the previous works often assume, SGD can exhibit many strange and potentially undesirable behaviors. Specifically, we construct landscapes and data distributions such that (1) SGD converges to local maxima, (2) SGD escapes saddle points arbitrarily slowly, (3) SGD prefers sharp minima over flat ones, and (4) AMSGrad converges to local maxima. We also realize results in a minimal neural network-like example. Our results highlight the importance of simultaneously analyzing the minibatch sampling, discrete-time updates rules, and realistic landscapes to understand the role of SGD in deep learning.

1 INTRODUCTION

SGD is the main optimization algorithm for training deep neural networks, and understanding SGD is widely regarded as a key step on the path to understanding deep learning (Bottou, 2012; Zhang et al., 2018; Xing et al., 2018; Mori et al., 2021; Du et al., 2018; Allen-Zhu et al., 2018; Wojtowytsch, 2021a;b; Gower et al., 2021; Ziyin et al., 2022b; Gurbuzbalaban et al., 2021; Zou et al., 2021; Li et al., 2021; Feng and Tu, 2021). Despite its algorithmic simplicity (could be described by only two lines of equations), SGD is hard to understand. The difficulty is threefold: (1) SGD is discrete-time in nature, and discrete-time dynamics are typically much more complicated to solve than their continuous-time counterparts (May, 1976); (2) SGD noise is state-dependent (Ziyin et al., 2022b; Hodgkinson and Mahoney, 2020); and (3) the loss landscape can be nonlinear and non-convex, involving many local minima, saddle points, and degeneracies (the Hessian is not full-rank) in the landscape (Xing et al., 2018; Wu et al., 2017). Each of these difficulties is so challenging that very few works attempt to deal with all of them simultaneously. Most previous works on SGD are limited to the cases when the loss landscape is strongly convex (Ziyin et al., 2022b; Liu et al., 2021; Hoffer et al., 2017; Mandt et al., 2017), or when the noise is assumed to be Gaussian and time-independent (Zhu et al., 2019; Jastrzebski et al., 2017; Xie et al., 2021); for the works that tackle SGD in a non-convex setting, often strong conditions are assumed. The reliance on strong assumptions regarding each of the challenges means that our present understanding of SGD for deep learning could be speculative. This work aims to examine some commonly held presuppositions about SGD and show that when all the three challenging factors are taken together, many counter-intuitive phenomena may arise. Since most of these phenomena are potentially undesired, this work can also be seen as a worst-case analysis of SGD.

In this work, we study the behavior of SGD in toy landscapes with non-convex loss and multi-minima. We approach the problem of SGD convergence from a different angle from many of the related works: instead of studying when SGD will converge, our result helps answer the question of when SGD might fail. In particular, the problem setting considers discrete-time SGD close to saddle points, where the noise is due to minibatch sampling, and the learning rate is held constant throughout training. In the next section, we define the SGD algorithm and the necessary notations. In Sec. 3, we discuss the related works. A warmup example is provided in Sec. 4. Sec. 5 introduces the main results. Sec. 6 presents the numerical simulations, including a minimal example involving a neural network. We also encourage the readers to examine the appendix. Sec. A presents additional experiments. Sec. B

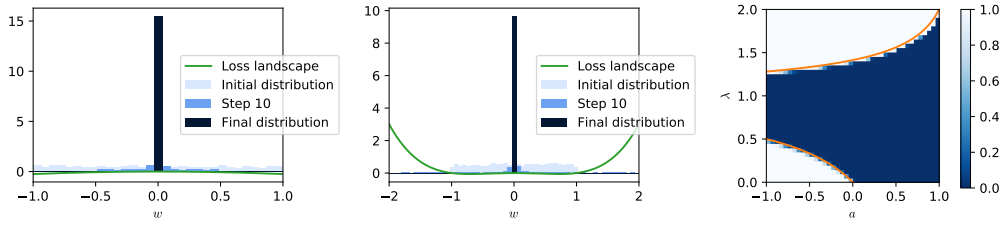

Figure 1: SGD converges to a local maximum for the example studied in Sec. 5.1. **Left**: Distribution of w on the quadratic potential, $\hat{L} = xw^2/2$; the histograms in different colors show the distribution of model parameters at different time steps. **Middle**: Distribution on the fourth-order potential, $\hat{L} = xw^2/2 + w^4/4$. **Right**: Escape probability as a function of a and λ. The parameters space is divided into an absorbing phase where w is attracted to the local maximum (in **dark blue**) and an active phase where w successfully escapes the two central bins (in white).

presents a continuous-time analysis of the problems in the main text and is relevant for discussing the unique discrete-time features of SGD. Sec. C presents all the proofs.

2 Background

Notation and Terminology. We use λ to denote the learning rate. w is the model parameter. $\hat{L}(w;x)$, with the caret symbol $\hat{\cdot}$, denotes the *sampled* loss function, which is a function of the data point, and $L := \mathbb{E}_x[\hat{L}]$ is the loss landscape averaged over the data distribution; for notational conciseness, we often hide the dependence of $\hat{L}$ on x when the context is clear. The lowercase t denotes the time step of optimization. The phrases *loss*, *objective*, and *potential* are also used interchangeably.

Now, we introduce the minibatch SGD algorithm. The objective of SGD can be defined as a pair $(\hat{L}, p(x))$ such that $\hat{L}$ is a differentiable function and $p(x)$ is a probability density. We aim at finding the minimizer of the following differentiable objective:

$$L(w) = \mathbb{E}_{x\sim p(x)}[\hat{L}(w;x)], \tag{1}$$

where x is a data point drawn from distribution $p(x)$ and $w \in \mathbb{R}^D$ is the model parameter. The gradient descent (GD) algorithm for L is defined as the update rule $w_t = w_{t-1} - \lambda \nabla_w L(w, x)$ for some randomly initialized w_0, where λ is the learning rate. The following definition defines SGD.

Definition 1. The *minibatch SGD* algorithm computes the update to the parameter w with the following equation: $w_t = w_{t-1} - \lambda \hat{g}_t$, where $\hat{g}_t = \nabla_{w_{t-1}} \hat{L}(w_{t-1}; x_t)$, and x_t is drawn from $p(x)$ such that x_i is independent of x_j for $i \neq j$.

3 Related Works

Escaping Saddle Points. Because neural networks are highly nonlinear, the landscape of neural networks is believed to have many local minima and saddle points (Du et al., 2017; Kleinberg et al., 2018; Reddi et al., 2018b). It is thus crucial for the optimization algorithm to be able to overcome saddle points efficiently and escape a suboptimal minimum. However, the majority of works on escaping saddle points assume no stochasticity or only artificially injected noise (Li et al., 2017; Du et al., 2017; Jin et al., 2017; Ge et al., 2015; Reddi et al., 2018b; Lee et al., 2016; Pemantle, 1990). Some physics-inspired approaches take the stochasticity into account but often rely on continuous-time approximation and nontrivial assumptions regarding the stationary distribution of the parameter (Mori et al., 2021; Xie et al., 2021; Liu et al., 2021). In the context of stochastic non-convex optimization, one setting that SGD is known to escape saddles and converge well is when the learning rate decreases to zero in the training (Pemantle, 1990; Vlaski and Sayed, 2019; Mertikopoulos et al., 2020). In contrast, we only consider the case when the learning rate is held fixed. We stress that our result does not contradict these previous results. On the opposite, our result reinforces the idea that the learning rate needs to be carefully chosen and scheduled when convergence is an essential concern.

Role of Minibatch Noise. One indispensable aspect of SGD is its stochasticity originated from minibatch sampling (Wu et al., 2020; Ziyin et al., 2022b; Zhu et al., 2019; Hodgkinson and Mahoney,

2020), whose strength and structure are determined by the model architecture and the data distribution. Moreover, it has been observed that the models trained with SGD significantly outperform the models directly trained with GD (Hoffer et al., 2017). Therefore, understanding the role of minibatch noise is of both fundamental and practical value. However, in the previous works, the focus is on analyzing the behavior of SGD on the landscape specified by L; the unique structures due to the noise in SGD are often treated in an oversimplified manner without referring to the actual models in deep learning or the actual data distributions. For example, Daneshmand et al. (2018) assumes that the noisy gradient is negatively correlated with the least-eigenvalue direction of the Hessian, an assumption that its validity is unclear when the Hessian is not full rank. Kleinberg et al. (2018) assumes that the noise in SGD makes the landscape one-point strongly convex, which implies that the loss landscape before the convolution is already close to a strongly convex landscape. Another commonly assumed condition is that the loss-landscape satisfies the Lojasiewicz condition, which is non-convex but implies the unrealistic condition that there is no local minimum except the global minimum (Karimi et al., 2020; Wojtowytsch, 2021a; Vaswani et al., 2019), which is not sufficient to understand the complicated dynamics that often happen in a setting with many minima (for example, a deep linear net (Ziyin et al., 2022a)). In Sec. 6.3, we show that these assumptions are violated for a minimal nonlinear neural network with two layers and one hidden neuron. While our example may or may not be directly relevant for realistic settings in the deep learning practice, our work highlights the importance of analyzing the actual noise structure imposed by a deep neural network in future works.

Large Learning Rate Regime. Recently, it has been realized that networks trained at a large learning rate have a better performance than networks trained with a vanishing learning rate (lazy training) (Chizat and Bach, 2018). Lewkowycz et al. (2020) shows that there is a qualitative difference between the lazy training regime and the large learning rate regime; the performance features two plateaus in testing accuracy in the two regimes, with the large learning rate regime performing much better. However, almost no previous theory exists for understanding SGD at a large learning rate. Our work is also relevant for understanding what happens at a large learning rate.

4 A Warm-up Example

This section studies a special example of the main results of our work. While the setting of this example is restricted, it captures the essential features and mechanisms of SGD that we will utilize to prove the main results. Consider the following loss function:

$$\hat{L}(w) = \frac{x}{2}w^2, \tag{2}$$

where $x, w \in \mathbb{R}$ and x is drawn from an underlying distribution $p(x)$ at every time-step. We let $\mathbb{E}[x] = a/2$ and assume that $\mathrm{Var}[x] = \sigma^2$ is finite. This gives rise to a true, sample-averaged loss of $L(w) = \frac{a}{4}w^2$. The stationary point of this loss function is $w = 0$. When $a > 0$, the underlying (deterministic) landscape is a minimum; the $a > 0$ case is now well understood in the discrete-time limit and when the underlying noise is state-dependent (Liu et al., 2021). When $a < 0$, the point $w = 0$ is a local maximum; one hopes that the underlying dynamics escape to infinity, and our goal is to understand the behavior of SGD in this case.

The following proposition and proof show that for the zero-mean and bounded distribution $p(x)$, SGD cannot escape from the local maximum when the learning rate is set to be 1.

Proposition 1. *Let $\lambda = 1$ and $a < 0$ and $p(x)$ be the distribution such that $x = 1$ and $x = -1 + a$ with equal probability. Then, w converges to $w = 0$ with probability 1 with the SGD algorithm, independent of the initialization.*

Proof. By the definition of the SGD algorithm, we have

$$w_{t+1} = \begin{cases} 0 & \text{with probability } 0.5; \\ (2-a)w_t & \text{otherwise.} \end{cases} \tag{3}$$

Therefore, after t time steps, w_t has at most 2^{-t} probability of being non-zero. For infinite t, $w_t = 0$ with probability 1. □

This simple toy example illustrates a few important points. First of all, it suggests that one cannot use the expected value of w to study the escape problem of SGD. The expectation value of w_t is

$$\mathbb{E}_x[w_t] = (1 - a/2)\mathbb{E}_x[w_{t-1}] = (1 - a/2)^t w_0 = w_0 e^{\ln(1-a/2)t}, \tag{4}$$

which is an exponentially fast escape from any initialization w_0 for $a < 0$. In fact, $\ln(1-a)$ is exactly the escape time scale of the GD algorithm. This is counter-intuitive: SGD will converge to the local maximum with probability 1, even if its expected value escapes the local maximum exponentially fast.

Alternatively, one might also hope to use the expected loss or the norm $\mathbb{E}_{w_t}[L(w_t)] \sim \|w_t\|_2^2$ as the metric of escape, but it is not hard to see that they suffer the same problem. At time t, because $w_t = 0$ with probability $1 - 2^{-t}$ and $w_t = (2-a)^t w_0$ with probability 2^{-t}, we have $\mathbb{E}[L(w_t)] \propto w_0^2 e^{2t \ln \frac{2-a}{\sqrt{2}}}$, which also predicts an exponentially fast escape. Again, this fails to reflect the fact that SGD has only a vanishingly small probability of escaping the constructed local maximum. Statistically speaking, the norms of w_t fail to be good metrics to measure the escape rate because these metrics are not robust against outliers. In our example, there is only 2^{-t} probability for SGD to escape the local minimum, yet the speed at which this outlier escapes the maximum outweighs the decay in its probability through time, thus contributing more to the expected norm than any other events. This example also shows the difficulty and subtlety of dealing with the escape problem. This example (together with Sec. 5.1) also emphasizes the importance of showing convergence in probability for future works in non-convex optimization because convergence in expectation is not sufficient to guarantee a good empirical performance. The results in the rest of this work can be seen as extensions of this special case to less restrictive settings.

5 Main Results

In this section, we present our four main theoretical results: (1) SGD may converge to a local maximum; (2) SGD may escape a saddle point arbitrarily slowly; (3) SGD may prefer sharp minima over flat ones; and (4) AMSGrad may converge to a local maximum.

5.1 SGD May Converge to a Local Maximum

We show that there are cases where SGD may fail to escape saddle points, while GD can successfully escape; this contradicts the suggestions in Kleinberg et al. (2018) that SGD always escapes a local minimum faster than GD. It is appropriate to begin with the definition of a saddle point.

Definition 2. w is said to be a stationary point of $L(w)$ if $\nabla_w L(w) = 0$. w is a minimum of L if, for all w' in a sufficiently small neighborhood of w, $L(w') \geq L(w)$. w is said to be a saddle point if w is a stationary point but not a local minimum.

This definition of saddle points, which includes local maxima, agrees with the common definition in the literature (Daneshmand et al., 2018). The following proposition generalizes our warmup example to a family of 1D problems in which SGD converges to a local maximum.

Proposition 2. *Let Eq. 2 be the loss function. Let $\lambda > 0$, and $p(x)$ be such that $\mu := \mathbb{E}[\ln|1-\lambda x|]$ and $s^2 := \mathrm{Var}[\ln|1-\lambda x|]$; then SGD converges to the local maximum in probability if $\mu < 0$.*

Proof Sketch. We first show that $\frac{1}{\sqrt{t}} \ln|w_t/w_0|$ obeys a normal distribution as $t \to \infty$. This allows us to deduce the cumulative distribution function (c.d.f.) of w as $t \to \infty$. The c.d.f. has an bifurcative dependence on the sign of $\mu := \mathbb{E}[\ln|1-\lambda x|]$. When $\mu < 0$, the distribution of w_t converges to 0 in probability. See Sec. C.2 for the detailed proof. □

Corollary 1. *Let Eq. 2 be the loss function and $p(x) = \frac{1}{2}\delta(x-1) + \frac{1}{2}\delta(x+1-a)$; then SGD converges to the local maximum in probability if*

$$\frac{a}{a-1} < \lambda < \frac{a - \sqrt{a^2 - 8a + 8}}{2(a-1)}. \tag{5}$$

One special feature of this example is that the state-dependent noise vanishes at the saddle point. In this particular example, this is achieved by having an SGD noise proportional to the gradient thus vanishing at the saddle point. This type of structure may be relevant for deep learning because vanishing noise at a stationary point and state-dependent noise can appear at the origin of a deep net, where weights of all layers are zero. Note that the convergence is independent of the actual shape of the distribution $p(x)$, and so may apply to many kinds of distributions besides the Bernoulli distribution we considered. There are three different regimes/phases for this simple escaping problem

(also see Fig. 1-Right). The escape regime $\lambda > \frac{a-\sqrt{a^2-8a+8}}{2(a-1)}$ is also present when $a > 0$; it means that in this regime, SGD will also escape w_0 even if it is a local minimum. This regime is not present if we perform a continuous-time analysis (Sec. B), showing that this region is due to the instability of the *discrete-time* SGD algorithm. This kind of escaping is undesirable because, in this regime, the local gradient cannot provide any guidance for minimizing the loss. The region $\frac{a}{a-1} < \lambda < \frac{a-\sqrt{a^2-8a+8}}{2(a-1)}$ is the trapped regime. This is due to the special structure of the minibatch noise – a comparison with the continuous-time analysis suggests that SGD will not be trapped if the noise is not position dependent (Sec. B.1). The small learning rate regime $\lambda < \frac{a}{a-1}$ is the successful escape regime because the SGD noise is of order λ^2 and becomes negligible at a small λ (Liu et al., 2021). This regime corresponds to the lazy learning regime of training: with negligible noise and vanishing gradient, SGD is known to optimize neural networks well (Du et al., 2018). This discussion also justifies our later choice for placing an upper bound to λ and focusing only on the second and third regime in Sec. 5.3 and 5.4.

It is important to understand why the previous works that show that the previous guarantees of convergence do not apply to this particular example, for example, Ge et al. (2015). In comparison, the SGD algorithm is not guaranteed to escape the saddle point in this example because two crucial assumptions of Ge et al. (2015) is violated: (1) the gradient does not have a bounded fluctuation around its mean (and so even parameters very far away from the saddle can be brought back close to the saddle point due to the noise), and (2) we do not inject additive noise to the gradient and the actual noise decreases as we approach the saddle point (and so the closer we are to the saddle point, the less likely one can escape). The standard theoretical guarantee of convergence in Ge et al. (2015) thus does not apply to this example. This example shows that the minibatch noise of SGD is very special and can strongly influence convergence. A continuous-time analysis in Section B of this example shows that the convergence to local maxima occurs when the minibatch noise dominates the gradient, namely, when the signal to noise ratio becomes smaller than 1. This suggests the importance of minibatch noise in influencing the dynamics of SGD. This suggests that the signal-to-noise ratio can be an important parameter in SGD dynamics and may deserve more future research.

5.2 SGD May Escape Saddle Points Arbitrarily Slowly

In the previous section, we have shown that one can adversarially construct a loss function for a fixed learning rate such that the SGD algorithm converges to a local maximum. This section shows that, even if one can tune the learning rate at will, there is a loss landscape where SGD can take an arbitrarily long time to escape, no matter how one chooses the learning rate. We begin by defining the escape rate.

Definition 3. The asymptotic average escape rate of w at learning rate λ, $\gamma(\lambda)$, is defined as $\gamma := \lim_{t\to\infty} \frac{1}{t}\mathbb{E}_{w_t}[\ln|w_t|]$. The optimal escape rate is defined as $\gamma^* := \sup_\lambda \gamma(\lambda)$.

Remark. *This definition can be seen as a generalization of the Lyapunov exponent to a probabilistic setting. We note that this definition of escape rate differentiates this work from the standard literature on escaping saddles, where the focus is on the time scale for reaching a local minimum when saddle points exist (Ge et al., 2015; Daneshmand et al., 2018; Jin et al., 2017), while our definition focuses on the time scale of escaping a specified saddle point. These two time scales can be vastly different.*

As discussed in the previous section, the undesired escape can be achieved by setting the learning rate to be arbitrarily large; however, this is not possible in practice because the learning rate of SGD is inherently associated with the stability of learning, and λ usually has to be much smaller than 1.

Proposition 3. *Let the loss function be Eq. (2), $a < 0$ and $p(x) \sim \frac{1}{2}\delta(x-1) + \frac{1}{2}\delta(x-a+1)$, where δ is the Dirac delta function, and $\lambda \le 1$. Then, for any $\epsilon > 0$, there exists a such that $\sup_\lambda \gamma(\lambda) < \epsilon$. Specifically, $\gamma^* = \ln\frac{2-a}{2\sqrt{1-a}}$, and, for any $\epsilon > 0$, $\gamma^* < \epsilon$ if $|a| \le 2\left|-e^\epsilon\sqrt{e^{2\epsilon}-1} - e^{2\epsilon} + 1\right|$.*

Namely, the SGD algorithm can escape the saddle point of Eq. (2) arbitrarily slowly. Intuitively, one expects escaping to be easier as we increase the learning rate (Kleinberg et al., 2018), and that the escape rate should monotonically increase as we increase the learning rate. This example conveys a novel message that the escaping efficiency is not monotonically increasing as the learning rate increases. For this example, the escape rate first decreases and starts to increase only when λ is quite large. This example also shows the subtlety in the escaping problem. On the one hand, one needs to avoid a too large learning rate to avoid training instability. On the other hand, one cannot use a too

small learning rate because a small learning rate also makes optimization slow. Thus, there should be a tradeoff between learning speed and learning stability, and we may construct a general theory for finding the best learning rate that achieves the best tradeoff; however, this tradeoff problem is a sufficiently complex problem on its own and is beyond the scope of the present work.

5.3 SGD May Prefer Sharper Minima

Modern deep neural networks are often overparametrized and can easily memorize all the training data points. This means that the traditional metrics such as Rademacher complexity cannot be used to guarantee the generalization capability of SGD (Zhang et al., 2017). Nevertheless, neural networks are found to generalize very well. This inspired the hypothesis that SGD, the main training algorithm of neural networks, contains some implicit regularization effect such that it biases the neural network towards simpler solutions (Neyshabur et al., 2017; Soudry et al., 2018). One hypothesized mechanism of this regularization is that SGD selects flat minima over the sharp ones (Hochreiter and Schmidhuber, 1997; Meng et al., 2020; Xie et al., 2021; Liu et al., 2021; Mori et al., 2021; Smith and Le, 2018; Wojtowytsch, 2021b). In this work, we show that this may not be the case because the dynamics and convergence of SGD crucially depend on the underlying mini-batch noise, while the sharpness of the landscape is independent of the noise. Before introducing the results, let us first define the sharpness of a local minimum.

Definition 4. Let $w = w^*$ be a local minimum of the loss function $L(w)$; the sharpness $s(w^*)$ of a local minimum w^* is defined as $s(w^*) := \mathrm{Tr}[\nabla_w^2 L(w^*)]$. We say that a minimum w_1^* is sharper than w_2^* if and only if $s(w_1^*) > s(w_2^*)$.

In this definition, the sharpness is the trace of the Hessian of the quadratic approximation to the local minimum. Other existing definitions, such as the determinant of the Hessian, are essentially similar to this definition (Dinh et al., 2017).

For an explicit construction, consider the following objective on a 2-dimensional landscape:

$$\hat{L}(w_1, w_2) = \frac{1}{2}\left[-(w_1 - w_2)^2 - (w_1 + w_2)^2 + (w_1 - w_2)^4 + (w_1 + w_2)^4 - 2aw_2^2 + xbw_1^2\right]. \quad (6)$$

for $a, b > 0$ and $p(x) = \frac{1}{2}\delta(x-1) + \frac{1}{2}\delta(x+1)$. The diagonal terms of the Hessian of the loss function is given by

$$\begin{cases} \frac{\partial^2}{\partial w_1^2}\hat{L}(w_1, w_2) = -2 + 12w_1^2 + 12w_2^2; \\ \frac{\partial^2}{\partial w_2^2}\hat{L}(w_1, w_2) = -2 - 2a + 12w_1^2 + 12w_2^2. \end{cases} \quad (7)$$

There are four local minima in total for this loss function. These minima and their sharpnesses are

$$\begin{cases} (w_1, w_2) = (\pm\frac{1}{\sqrt{2}}, 0) \quad \text{with } s = 4; \\ (w_1, w_2) = (0, \pm\sqrt{\frac{1+a}{2}}) \quad \text{with } s = 4 + 4a. \end{cases} \quad (8)$$

We see that for positive a, the minima at $(0, \pm\sqrt{\frac{1+a}{2}})$ are sharper than the other two minima. We show that SGD will converge to these sharper minima. For this example, we assume that w_1 and w_2 are bounded because, if initialized sufficiently far from the origin, the fourth-order term will cause divergence.

Proposition 4. *Let the loss function be Eq. (6), $|w_1| \le 1$ and $|w_2| \le 1$. For any $\lambda < c$ for some constant $c = O(1)$, there exists some b such that if SGD converges in probability, it will converge to the sharper minimum at $(w_1, w_2) = (0, \pm\sqrt{\frac{1+a}{2}})$.*

Remark. *In this example, the noise is not full-rank, which is often the case in a deep learning setting for overparametrized networks (Wojtowytsch, 2021a). For example, when weight decay is used, the Hessian of the loss function should be full rank, while the rank of the noise covariance should be proportional to the inherent dimension of the data points, which is in general much smaller than the dimension of the Hessian in deep learning (Ansuini et al., 2019).*

The proof is technical and given in the appendix Sec. C.5. In fact, it has been controversial whether finding a flat minimum can help generalization. For example, Dinh et al. (2017) shows that, for every flat minimum of a ReLU-based net, there exists a minimum that is arbitrarily sharper and generalizes

as well. However, this work does not rule out the possibility that conditioning by using gradient-based optimization, the performance of the sharper minima that gradient descent finds is worse than the performance of the flatter minima. If this assumption is valid, then biasing gradient descent towards flatter minima can indeed help, and the stochasticity of SGD has been hypothesized to help in this regard. However, our construction shows that SGD, on its own, may be incapable of helping SGD converge to a flatter minimum. At least some other assumptions need to be invoked to show that SGD may help. For example, the definition of the neural networks may endow these models with a special kind of Hessian and noise structure that, when combined with SGD, results in a miraculous generalization behavior. However, no previous work has pursued this direction in sufficient depth to our knowledge, and future studies in this direction may be fruitful.

5.4 Non-Convergence of Adaptive Gradients

Adam (Kingma and Ba, 2014) and its closely related variants such as RMSProp (Tieleman and Hinton, 2012) have been shown to converge to a local maximum even for some simple convex loss landscapes (Reddi et al., 2018a). The proposed fix, named AMSGrad, takes the maximum of all the previous preconditioners in Adam. This section shows that AMSGrad may also converge to a local maximum even in simple non-convex settings. Let $x_t \sim p(x)$ and $\hat{g}_t = \nabla \hat{L}(x_t, w_{t-1})$, the Adam algorithm is given by Eq. (9)-(12), where $v_0 = m_0 = 0$. We have removed the numerical smoothing constant ϵ from the denominator, which causes no problem if w_0 is initialized away from 0. Here, β_1 is the momentum hyperparameter. v_t is called the preconditioner, and β_2 is the associated hyperparameter. The standard value for β_1 is 0.9 and β_2 is 0.999. In our theory, we only consider the case when $\beta_1 = 0$. The experiment section shows that a similar problem exists when $\beta_1 > 0$ (with additional interesting findings). Intuitively, this is easy to understand because AMSGrad behaves like SGD asymptotically, so we only have to wait for long enough, and the results in previous sections would apply. The construction below follows this intuition.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\hat{g}_t; \quad (9)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\hat{g}_t^2; \quad (10)$$

$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t); \quad (11)$$

$$w_t = w_{t-1} - \frac{\lambda}{\sqrt{\hat{v}_t}} m_t. \quad (12)$$

Proposition 5. *Let $w_t \in [-1, 1]$, and $w_0 \neq 0$. For fixed $\lambda < 1$ and the loss function in Eq. (2) there exists $a < 0$ such that the AMSGrad algorithm converges in probability to a local maximum.*

Remark. *The proof is given in Sec. C.6. Note that the example we construct is similar to the original construction in Reddi et al. (2018a) that shows that Adam converges to a maximum while AMSGrad succeeds in reaching the global minimum. In their example (with some rescaling), the gradient is a random variable*

$$\hat{g}_t = \begin{cases} 1 & \text{with probability } q \approx 1; \\ c_0 < 0 & \text{with probability } 1-q \ll 1; \end{cases} \quad (13)$$

such that the expected gradient is negative, and Adam is shown to converge to the direction opposite to the gradient descent (therefore to a maximum). In our example, the gradient is (roughly)

$$\hat{g}_t = \begin{cases} w_{t-1} & \text{with probability } 1/2; \\ -(1+a)w_{t-1} & \text{with probability } 1/2. \end{cases} \quad (14)$$

Therefore, our example can be seen as a minimal generalization of the Reddi et al. (2018a) example to a non-convex setting.

6 Experimental Demonstrations

We perform experiments to illustrate the examples studied in this work. Due to space limitations, we illustrate the escape rate and the convergence to sharper minima example in Appendix Sec. A.

6.1 SGD Converges to a Local Maximum

Numerical results are obtained for the setting described in section 5.1. See Fig. 1-Left. The loss landscape is defined by $L(w) = \frac{1}{4}aw^2$. In this numerical example, we set $\lambda = 0.8$ and $a = -1$, and the histogram is plotted with 2000 independent runs. We see that the distribution converges to the local maximum at $w = 0$ as the theory predicts. For better qualitative understanding, we also plot

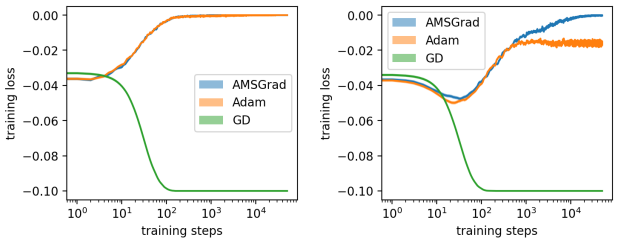

Figure 2: AMSGrad diverges to the local maximum with or without momentum in the example we studied in Sec 5.4. Our result shows that AMSGrad is always attracted to the local maximum, while Adam has the potential of escaping the local maximum with momentum. **Left**: without momentum. **Right**: with momentum.

the empirical phase diagram of this setting in Fig. 1-Right. We perform numerical evaluations for different combinations of a and λ on a $a - \lambda$ plane, and for each combination, we plot the percentage of w that escapes the central bin (white is 100% and **dark blue** is 0%). For completeness, we also plot the case when $a > 0$, i.e., when the critical point at $w = 0$ is a local minimum. The orange line shows where the phase transition is expected to happen according to Eq. (5). The fact that the orange line agrees exactly with the empirical phase transition line confirms our theory.

We also numerically study a closely related fourth-order loss landscape, defined as $\hat{L}(w) = \frac{1}{2}xw^2 + \frac{1}{4}w^4$, where the distribution $p(x)$ is the same as before. The expected loss $L(w)$ has two local minima located as $w = \pm\sqrt{-2a}$. We are interested in whether SGD can converge to these two points successfully; note that, without the fourth-order term, this loss is the same as the quadratic loss we studied. See Fig. 1-Middle. As before, $a = -1$ and $\lambda = 0.8$. We see that the distribution of w also concentrates towards the local maximum after training, and no w is found to converge to the two local minimum even if a significant proportion of w is initialized close to these two minima. A phase diagram analysis is given in the Sec. A.1.

6.2 Non-Convergence of AMSGrad

In this section, we illustrate the example of the convergence behavior of AMSGrad described in Sec. 5.4; for reference, we also plot the behavior of Adam and gradient descent for this example. See Fig. 2. The experiment is the average over 2000 runs, each with 5×10^4 steps. The uncertainty is reflected by the shaded region (almost invisibly small). The loss function is the same as discussed in Sec. 5.4 with $a = -0.1$. For illustration purposes, we set $\lambda = 0.2$ and $\beta_2 = 0.999$ for both Adam and AMSGrad. When momentum is used, we set $\beta_1 = 0.9$. GD is run with a learning rate of 0.01. Without momentum, both Adam and AMSGrad converge to the local maximum with almost the same speed. When momentum is added, AMSGrad still converges to the local maximum. Curiously, Adam is no longer attracted to the local maximum but also remains away from the local minimum. This suggests that Adam, with momentum, might have a better capability of escaping saddle points than AMSGrad and might be preferable to AMSGrad in a non-convex setting.

6.3 A Toy Neural Network Example

We have now illustrated several SGD phenomena we studied in artificial settings, inspiring the natural question of whether they occur for an actual neural network. Here we construct a neural-net-like optimization problem to show that the problems with convergence might indeed arise for a neural network. The toy neural network function is given by $f(x) = w_2\sigma(w_1 x)$, where $w_1, w_2 \in \mathbb{R}$. σ is the nonlinearity, and we let $\sigma(x) = x^2$ as a minimal example. This is the simplest kind of nonlinear feedforward network one can construct (2 layer with a single hidden neuron). We also pick a minimal dataset with a single data point: $x = 1$ with probability 1 and $y \in \{-1, 2\}$, each with probability 0.5, i.e., the label has a degree of inherent uncertainty, which is often the case in real problems. We use mean squared error (MSE) as the loss function. Therefore, the expected loss is $L(w_1, w_2) = \frac{1}{2}(w_2 w_1^2 + 1)^2 + \frac{1}{2}(w_2 w_1^2 - 2)^2$. The global minimum $L^* = 2.25$ is degenerate, and is achieved when $w_1 = 1/\sqrt{2w_2}$. There is also a manifold of saddle points given by $\{(w_1, w_2)|w_1 = 0, w_2 \geq 0\}$, all with $L = 2.5$. Despite the simplicity of this example, it contains a few realistic features in a realistic problem, including (1) inherent uncertainty in the data, (2) a nonlinear hidden layer, and (3) a degenerate minimum with zero eigenvalues in the Hessian.

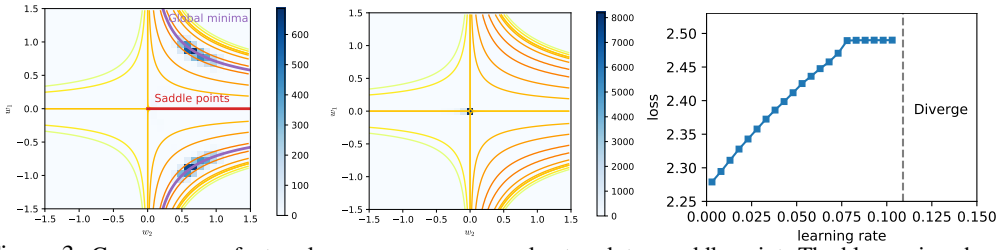

Figure 3: Convergence of a two-layer one-neuron neural network to a saddle point. The blue region shows the empirical density of converged parameter distribution. **Left**: $\lambda = 0.001$ at step 10000 converges to global minima. **Mid**: $\lambda = 0.1$ at step 10000 converges to a saddle point. **Right**: Average loss in equilibrium as a function of learning rate. The loss function diverges for learning rates larger than 0.108.

More importantly, most previous works on the convergence or escaping behavior of SGD are inapplicable to this example. One dominant assumption is the ρ-Hessian Lipschitz property (Jin et al., 2017; Ge et al., 2015), which does not hold for this example in particular. Another common assumption is the PL condition (Karimi et al., 2020; Wojtowytsch, 2021a; Vaswani et al., 2019), which does not apply due to the existence of the saddle point. One recent assumption is that the loss function is 1-point convex for all points (Kleinberg et al., 2018), which is also ruled out because $(0, 0)$ is not 1-point strongly convex. Another relevant assumption is the correlated negative curvature assumption, which also does not hold for $(0, 0)$. We explain in Sec. A.5 in detail why these conditions are violated.

See Fig. 3 for the experimental results with this example. Here, w_1 is initialized uniformly in $[-1, 1]$; w_2 is initialized uniformly in $[0, 1]$ to be closer to the global minimum and away from the saddle point (standard initialization such as initializing in $[-1, 1]$ does not change the conclusion). The left figure shows the stationary distribution of the model parameter at a small learning rate ($\lambda = 0.001$). All the parameters are located in the global-minimum valley as expected. In contrast, when the learning rate is large ($\lambda = 0.1$), the central figure shows that all models (1000 independent runs) converge to the saddle point at $(0, 0)$. The right figure investigates this change more systematically and shows the change in the average stationary training loss of the models as we increase the learning rate from 0.001 to 0.15. We see that for a small learning rate, the training loss is close to that of the global minima ($L = 2.25$), and for a significant range of large learning rates, the model invariably converges to a saddle point ($L = 2.5$). One additional interesting observation is that the model diverges at $\lambda \approx 0.11$, and there is almost no sign of such divergence (such as increased fluctuation) when the learning rate is close to the divergent threshold. This example shows the relevance of our results to the study of neural networks. In fact, it has often been noticed that at convergence, many large-scale neural networks in real tasks exhibit negative eigenvalues in the Hessian (Alain et al., 2019; Granziol et al., 2019). The existence of negative eigenvalues at a late time suggests either the possibility that the algorithm has yet to escape or that it has indeed been attracted to such saddle points; if the latter is true, then our work offers an explanation. The emergence of negative eigenvalues at the end of training serves as indirect evidence that our result may be relevant for larger neural networks. It thus remains an important open question to prove (and identify the condition of) or disprove the convergence of larger and deeper neural networks to a local maximum.

7 Discussion

We have shown that when the learning rate is not carefully chosen or scheduled, SGD can exhibit many undesirable behaviors, such as convergence to local maxima or saddle points. The limitation of our work is clear. At best, all the constructions we made are minimal and simplistic toy examples that are far from practice, and investigating whether the discovered messages are relevant for deep learning or not is the one important immediate future step. Other relevant questions include: (1) Does convergence to saddle points help or hurt generalization? (2) If it hurts, how can we modify SGD to avoid saddle points better? We suspect changing learning rates, changing batch size, or injecting noise may help, but a convincing theoretical guarantee in a realistic setting is yet lacking. (3) If saddle points do not have worse generalization, our results motivate for understanding why. This is especially relevant because we showed that using a large learning rate is more likely to converge to saddle points, while previous works have shown that using a large learning rate can improve generalization; combined, this may imply that certain saddle points may have intriguing but unknown regularization effects.

ACKNOWLEDGMENT

We thank the anonymous reviewers for providing detailed and constructive feedback for our draft. Ziyin thanks Jie Zhang for the help during the writing of this manuscript. Ziyin is financially supported by the GSS Scholarship from the University of Tokyo. BL acknowledges CNRS for financial support and Werner Krauth for all kinds of help. JS gratefully acknowledges support from the National Science Foundation Graduate Fellow Research Program (NSF-GRFP) under grant DGE 1752814. This work was supported by KAKENHI Grant Numbers JP18H01145 and JP21H05185 from the Japan Society for the Promotion of Science.

REFERENCES

Alain, G., Roux, N. L., and Manzagol, P.-A. (2019). Negative eigenvalues of the hessian in deep neural networks. *arXiv preprint arXiv:1902.02366*.

Allen-Zhu, Z., Li, Y., and Song, Z. (2018). A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*.

Ansuini, A., Laio, A., Macke, J. H., and Zoccolan, D. (2019). Intrinsic dimension of data representations in deep neural networks.

Bottou, L. (2012). Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, pages 421–436. Springer.

Chizat, L. and Bach, F. (2018). A note on lazy training in supervised differentiable programming. *arXiv preprint arXiv:1812.07956*, 8.

Daneshmand, H., Kohler, J., Lucchi, A., and Hofmann, T. (2018). Escaping saddles with stochastic gradients. In *International Conference on Machine Learning*, pages 1155–1164. PMLR.

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. (2017). Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pages 1019–1028. PMLR.

Du, S. S., Jin, C., Lee, J. D., Jordan, M. I., Poczos, B., and Singh, A. (2017). Gradient descent can take exponential time to escape saddle points. *arXiv preprint arXiv:1705.10412*.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. (2018). Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*.

Feng, Y. and Tu, Y. (2021). The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima. *Proceedings of the National Academy of Sciences*, 118(9).

Ge, R., Huang, F., Jin, C., and Yuan, Y. (2015). Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory*, pages 797–842. PMLR.

Gower, R., Sebbouh, O., and Loizou, N. (2021). Sgd for structured nonconvex functions: Learning rates, minibatching and interpolation. In *International Conference on Artificial Intelligence and Statistics*, pages 1315–1323. PMLR.

Granziol, D., Garipov, T., Zohren, S., Vetrov, D., Roberts, S., and Wilson, A. G. (2019). The deep learning limit: are negative neural network eigenvalues just noise? In *ICML 2019 Workshop on Theoretical Physics for Deep Learning*.

Gurbuzbalaban, M., Simsekli, U., and Zhu, L. (2021). The heavy-tail phenomenon in sgd. In *International Conference on Machine Learning*, pages 3964–3975. PMLR.

Hochreiter, S. and Schmidhuber, J. (1997). Flat minima. *Neural Computation*, 9(1):1–42.

Hodgkinson, L. and Mahoney, M. W. (2020). Multiplicative noise and heavy tails in stochastic optimization. *arXiv preprint arXiv:2006.06293*.

Hoffer, E., Hubara, I., and Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1729–1739.

Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. (2017). Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*.

Jin, C., Ge, R., Netrapalli, P., Kakade, S. M., and Jordan, M. I. (2017). How to escape saddle points efficiently. In *International Conference on Machine Learning*, pages 1724–1732. PMLR.

Karimi, H., Nutini, J., and Schmidt, M. (2020). Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kleinberg, B., Li, Y., and Yuan, Y. (2018). An alternative view: When does sgd escape local minima? In *International Conference on Machine Learning*, pages 2698–2707. PMLR.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. (2016). Gradient descent only converges to minimizers. In *Conference on learning theory*, pages 1246–1257. PMLR.

Lewkowycz, A., Bahri, Y., Dyer, E., Sohl-Dickstein, J., and Gur-Ari, G. (2020). The large learning rate phase of deep learning: the catapult mechanism. *arXiv preprint arXiv:2003.02218*.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2017). Visualizing the Loss Landscape of Neural Nets. *ArXiv e-prints*.

Li, Z., Malladi, S., and Arora, S. (2021). On the validity of modeling sgd with stochastic differential equations (sdes). *arXiv preprint arXiv:2102.12470*.

Liu, K., Ziyin, L., and Ueda, M. (2021). Noise and fluctuation of finite learning rate stochastic gradient descent.

Mandt, S., Hoffman, M. D., and Blei, D. M. (2017). Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18:1–35.

May, R. (1976). Simple mathematical models with very complicated dynamics. nature. vol. 251, june. 10.

Meng, Q., Gong, S., Chen, W., Ma, Z.-M., and Liu, T.-Y. (2020). Dynamic of stochastic gradient descent with state-dependent noise. *arXiv preprint arXiv:2006.13719*.

Mertikopoulos, P., Hallak, N., Kavis, A., and Cevher, V. (2020). On the almost sure convergence of stochastic gradient descent in non-convex problems. *arXiv preprint arXiv:2006.11144*.

Mori, T., Ziyin, L., Liu, K., and Ueda, M. (2021). Logarithmic landscape and power-law escape rate of sgd. *arXiv preprint arXiv:2105.09557*.

Neyshabur, B., Tomioka, R., Salakhutdinov, R., and Srebro, N. (2017). Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*.

Pemantle, R. (1990). Nonconvergence to unstable points in urn models and stochastic approximations. *The Annals of Probability*, 18(2):698–712.

Reddi, S., Kale, S., and Kumar, S. (2018a). On the convergence of adam and beyond. In *International Conference on Learning Representations*.

Reddi, S., Zaheer, M., Sra, S., Poczos, B., Bach, F., Salakhutdinov, R., and Smola, A. (2018b). A generic approach for escaping saddle points. In *International conference on artificial intelligence and statistics*, pages 1233–1242. PMLR.

Smith, S. L. and Le, Q. V. (2018). A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. (2018). The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Vaswani, S., Bach, F., and Schmidt, M. (2019). Fast and faster convergence of sgd for over-parameterized models and an accelerated perceptron. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1195–1204. PMLR.

Vlaski, S. and Sayed, A. H. (2019). Second-order guarantees of stochastic gradient descent in non-convex optimization.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer.

Wojtowytsch, S. (2021a). Stochastic gradient descent with noise of machine learning type. part i: Discrete time analysis. *arXiv preprint arXiv:2105.01650*.

Wojtowytsch, S. (2021b). Stochastic gradient descent with noise of machine learning type. part ii: Continuous time analysis. *arXiv preprint arXiv:2106.02588*.

Wu, J., Hu, W., Xiong, H., Huan, J., Braverman, V., and Zhu, Z. (2020). On the noisy gradient descent that generalizes as sgd. In *International Conference on Machine Learning*, pages 10367–10376. PMLR.

Wu, L., Zhu, Z., and E, W. (2017). Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes. *ArXiv e-prints*.

Xie, Z., Sato, I., and Sugiyama, M. (2021). A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*.

Xing, C., Arpit, D., Tsirigotis, C., and Bengio, Y. (2018). A walk with sgd. cite arxiv:1802.08770Comment: First two authors contributed equally.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2017). Understanding deep learning requires rethinking generalization.

Zhang, C., Liao, Q., Rakhlin, A., Miranda, B., Golowich, N., and Poggio, T. (2018). Theory of deep learning iib: Optimization properties of sgd. *arXiv preprint arXiv:1801.02254*.

Zhu, Z., Wu, J., Yu, B., Wu, L., and Ma, J. (2019). The anisotropic noise in stochastic gradient descent: Its behavior of escaping from sharp minima and regularization effects. In *International Conference on Machine Learning*, pages 7654–7663. PMLR.

Ziyin, L., Li, B., and Meng, X. (2022a). Exact solutions of a deep linear network. *arXiv preprint arXiv:2202.04777*.

Ziyin, L., Liu, K., Mori, T., and Ueda, M. (2022b). Strength of minibatch noise in SGD. In *International Conference on Learning Representations*.

Zou, D., Wu, J., Braverman, V., Gu, Q., and Kakade, S. M. (2021). Benign overfitting of constant-stepsize sgd for linear regression. *arXiv preprint arXiv:2103.12692*.

A ADDITIONAL EXPERIMENTS

A.1 PHASE DIAGRAM OF THE FOURTH-ORDER LOSS FUNCTION

With two proper local minima, the fourth-order loss function is a more realistic loss function than the one we considered in Sec. 5.1. We also performed one experiment in the main text (See Fig. 1). Here, we plot its empirical phase diagram in Figure 4. We see that for this loss landscape also, there is some region such that for all λ, SGD converges to the local maximum. This loss landscape is very difficult to study in discrete time as it is known to lead to chaotic behavior at large learning rate (May, 1976). Therefore, we alternatively try to understand this landscape using continuous approximation. See Sec. B.

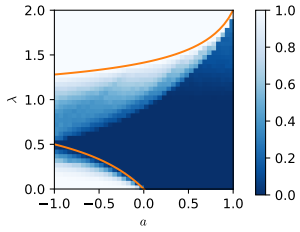

Figure 4: Escape probability from the local maximum as a function of a and λ with fourth-order loss landscape. The parameters space is divided into an absorbing phase where w is attracted to the local maximum and an active phase where w escapes to infinity successfully. The orange line is the theoretical phase transition line for the quadratic loss function. We see that when λ is small, the line based on quadratic loss also gives good agreement with the fourth-order loss. This suggests that part of this result is universal and independent of the details of the loss function.

A.2 ESTIMATING THE ESCAPE RATE

In the escape rate experiments, the empirical results are obtained from the proposed approximation. Here, we show that it is valid. See Fig. 5, where we plot the estimated γ as a function of the training step. We see that the estimated value converges within about 20 steps. We, therefore, estimate the escape rates at time step 100 in the main text.

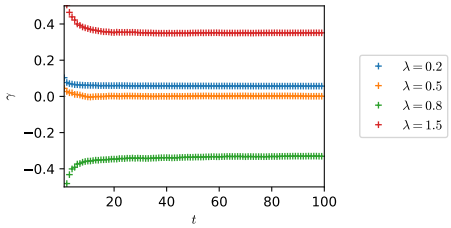

Figure 5: Escape rate γ as a function of time step t obtained when $a = -1$, showing that γ, defined in Equation (73), is indeed a well-defined quantity when t is large. γ becomes stable after roughly 40 steps.

A.3 ESCAPE RATE EXPERIMENTS

This section illustrates the slow escape problem studied in Sec. 5.2. See Fig. 6. γ is calculated by averaging the first 50 time steps across 2000 independent runs. We see that, as our theory predicts, as $|a|$ gets closer to 0, the optimal escape rate decreases towards 0. This implies that there exists a landscape such that SGD is arbitrarily slow at learning, independent of parameter tuning. One additional observation is that the optimal learning rate is neither too large nor too small, i.e., there seems to be a tradeoff between the escape speed and escape probability (and between the speed and stability). See also the discussion at the end of Sec. 5.2.

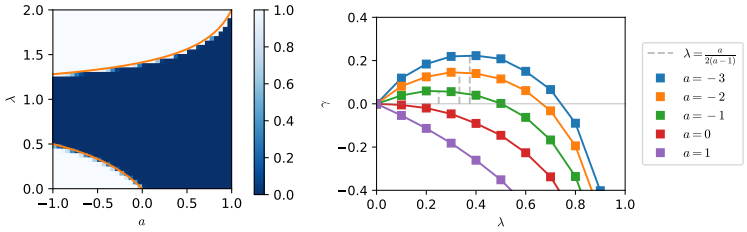

Figure 6: Escape rate γ as a function of learning rate λ with quadratic loss landscape, obtained by simulations. We note that the escape-rate analysis yields results which are compatible with the phase diagram.

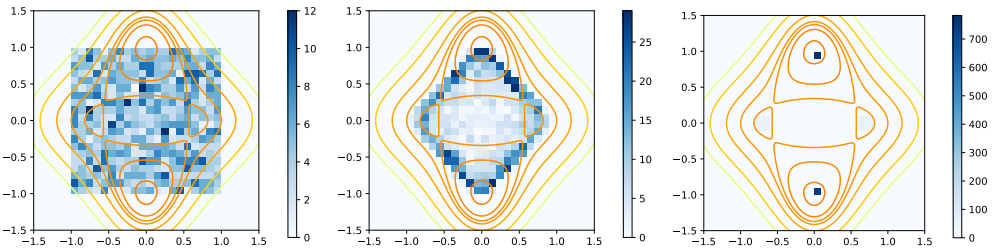

Figure 7: Evolution of the distribution of w in a landscape where two flat minima and two sharp minima exist. As Proposition 4 shows, the model parameters converge to the sharper minima even if initialized in the flat minimum. **Left**: Initialization. **Mid**: step 2. **Right**: step 10000.

A.4 Convergence to the Sharper Minimum

We use the same 2-dimensional loss landscape defined in Sec. 5.3. The experiment is run with 2000 independent simulations and learning rate $\lambda = 0.05$. See Fig. 7, where we overlap the underlying landscape with the empirical distribution (the heat map). We see that, even though the initialization overlaps significantly with the local flatter minimum, all points converge to the sharper minimum, as the theorem predicts.

A.5 Inadequacy of Standard Assumptions for the Neural Network Example

In this section, we show that the standard assumptions in the previous literature do not hold for the example studied in Sec. 6.3. One common assumption is that the loss function is ρ-Hessian Lifschitz.

Definition 5. A loss function L is said to be ρ-Hessian Lifschitz if, for some $\rho > 0$,

$$\forall \mathbf{w}, \mathbf{w}', \|\nabla^2 L(\mathbf{w}) - \nabla^2 L(\mathbf{w})\| \leq \rho \|\mathbf{w} - \mathbf{w}'\|. \tag{15}$$

This certainly does not hold for the example we considered. Explicitly, let $w_2 = 0$, the Hessian $H(\mathbf{w})$ of the loss function is

$$H_{w_1,w_1} = H_{w_1,w_2} = H_{w_2,w_1} = 0, \tag{16}$$

and

$$H_{w_2,w_2} = -w_1^2. \tag{17}$$

Let $w_1 \to \infty$ violates the ρ-Hessian Lifschitz property.

Another common assumption for the non-convex setting is the PL condition.

Definition 6. Let L^* be the value of L at the global minimum. A loss function L is said to satisfy the PL (Lojasiewitz) condition if

$$\forall \mathbf{w}, \|\nabla L(\mathbf{w})\|^2 \geq \mu(L(\mathbf{w}) - L^*), \tag{18}$$

for some $\mu > 0$.

This also does not hold due to the existence of the saddle point. Let $(w_1, w_2) = (0, 0)$. The gradient is zero, but the right-hand side is greater than 0.

Recently, the correlated negative curvature assumption has been proposed to study the escaping behavior of SGD.

Definition 7. Let $\mathbf{v_w}$ be the eigenvector of the minimum eigenvalue of the Hessian $H(\mathbf{w})$. L satisfies the correlated negative curvature assumption if, for some $\gamma > 0$,

$$\forall \mathbf{w}, \mathbb{E}_x[\langle \mathbf{v_w}, \nabla L(x, \mathbf{w})\rangle^2] > \gamma \tag{19}$$

where x is the data point.

This also does not hold. The point $(0, 0)$ violates this condition because $\nabla L(x, \mathbf{w}) = 0$ for all x at $(w_1, w_2) = (0, 0)$. In fact, it is exactly this point that violates this condition that the SGD converges to in this example.

Recently, the one point strongly convex assumption has been proposed to study the escaping behavior of SGD.

Definition 8. A loss function is said to satisfy the c-one point strongly convex condition with respect to $\mathbf{w}^*$ for all gradient noise ϵ, $c > 0$, define $\mathbf{v} = \mathbf{w} - \lambda \nabla L(\mathbf{w})$, we have

$$\langle -\nabla \mathbb{E}_\epsilon L(\mathbf{v} - \lambda \mathbf{w}, \mathbf{w}^* - y) \geq c||\mathbf{w}^* - \mathbf{v}||_2^2, \tag{20}$$

where ϵ is the random noise caused by minibatch sampling.

This assumption is equivalent to that the loss landscape is strongly convex after convoluting with the noise. This condition implies that there is only a single stationary point. However, this is not the case, consider any point with $w_1 = 0$ and $w_2 \leq 0$. These points have zero gradient for $w_1 \leq 0$ and so $\epsilon = 0$ with probability 1. Then, these points all have the same loss after the convolution:

$$\mathbb{E}[L(\mathbf{v} - \lambda \epsilon)] = L(\mathbf{v}), \tag{21}$$

which either imply that the landscape is not convex or that there is more than 1 stationary point, and so the system is not one point strongly convex [1].

[1] One notices that the origin, where all the parameters are zero, is a special point in the problem. In fact, the origin may be a very special point in the landscape of deep neural networks in general. For example, see Ziyin et al. (2022a).

B CONTINUOUS-TIME APPROXIMATION WITH FOKKER-PLANCK EQUATION

In this section, we study the examples we studied in the main text in a continuous-time approximation, in order to understand the unique aspects of discrete-time SGD. Compared with the discrete-time analysis, continuous-time analysis is usually more powerful in terms of calculation: it is able to deal with more complicated potentials.

B.1 FOKKER-PLANCK EQUATION AND ITS STATIONARY DISTRIBUTION

Recall that the SGD update takes the form (see Sec. 2)

$$\Delta w_t = -\lambda \nabla L + \lambda \sqrt{C} \eta_t, \tag{22}$$

when $\lambda < 1$, the above equation may be approximated by a continuous-time Ornstein-Uhlenbeck process (Mandt et al., 2017)

$$dw(t) = -\lambda \nabla L dt + \frac{\lambda}{\sqrt{S}} \sqrt{C(w)} dW(t), \tag{23}$$

where λ is the learning rate; we have also introduced S as the batch size. $dW(t)$ is a stochastic process satisfying

$$\begin{cases} dW(t) \sim \mathcal{N}(0, dtI), \\ \mathbb{E}[dW(t) dW(t')] \propto \delta(t - t'). \end{cases} \tag{24}$$

By the definition of the underlying discrete-time process, the term $dw(t)$ depends only on $w(t)$ and has no dependence on $w(t + dt)$. Thus, the stochastic integration should be interpreted as Ito.

For simplicity, we only consider one-dimensional version of the Fokker-Planck, which is

$$\frac{\partial P[w, t|w(0), 0]}{\partial t} = -\frac{\partial}{\partial w} J(w, t|w(0), 0), \tag{25}$$

where $J[w, t|w(0), 0]$ is the probability flow. The current $J[w, t|w(0), 0]$ is:

$$J[w, t|w(0), 0] = \lambda \frac{\partial L}{\partial w} P[w, t|w(0), 0] + \frac{\lambda^2}{2S} \frac{\partial}{\partial w} \{C(w) P[w, t|w(0), 0]\}. \tag{26}$$

Assuming that the SGD dynamics is ergodic, there is a unique stationary distribution for w when $t \to \infty$ that can be found to be

$$P(w) \propto \frac{1}{C(w)} \exp\left[-\frac{2S}{\lambda} \int dw \frac{1}{C(w)} \frac{\partial L}{\partial w} \right]. \tag{27}$$

We will apply this equation to study the relevant problems in this work.

B.2 A GENERAL CASE

In this section, we consider (a slightly more general version of) the fourth-order potential we studied in the main text. The average loss landscape in this model is

$$L(w) = \frac{1}{4} a w^2 + \frac{1}{4} b w^4, \tag{28}$$

where $b > 0$, guarantees that the w is bounded regardless the value of a. In the limit of $b \to 0$, this function reduces to saddle point problem we studied in Sec. 5.1. Besides the SGD noise, additive noise is also present in the dynamics. The variance of the additive noise has no dependence on w. When the additive noise is present, the update rule (equation of motion) is

$$dw(t) = -[\lambda(a/2) w(t) + b w^3(t)] dt + \lambda \eta_m(t) + \lambda \eta_a(t), \tag{29}$$

where both $\eta_m(t)$ and $\eta_a(t)$ are both Ornstein-Uhlenbeck process, denoting multiplicative noise and additive noise respectively. The w-dependent covariance of w is $C(w) = w^2$. The SGD noise is thus a multiplicative noise. $\eta_m(t)$ and has a variance $\frac{w(t)^2}{S} dt$, while the additive noise $\eta_a(t)$ has a variance of $\sigma^2 dt$, where σ is a positive constant denoting the strength of the additive noise. If the

multiplicative noise is seen as a part of the loss landscape, (2) coincides with the 2nd-order term in the loss landscape. There is no correlation between the additive noise and the SGD noise, i.e. $\mathbb{E}[\eta_m(t)\eta_a(t')] = 0$. The additive noise can be seen as the artificially injected noise, a technique sometimes used for aiding the escape or for Bayesian learning purposes (Jin et al., 2017; Welling and Teh, 2011).

The additive noise vanishes in the limit of $\sigma \to 0$, and the model becomes a normal SGD with 4th-order loss function. It is always possible to define a noise $\eta'(t)$, whose contribution is equivalent to the contribution of both the SGD and the additive noise, and the equation of motion becomes

$$dw(t) = -[\lambda(a/2)w(t) + bw^3(t)]dt + \lambda\eta'(t). \tag{30}$$

This transformed noise $\eta'(t)$ thus has 0 mean and a variance of $\left(\frac{w(t)^2}{S} + \sigma^2\right)dt$. Define $\eta(t) = \eta'(t)/\sqrt{\frac{w(t)^2}{S} + \sigma^2}$, the equation of motion becomes

$$dw(t) = -\lambda[(a/2)w(t) + bw^3(t)]dt + \lambda\sqrt{\frac{w(t)^2}{S} + \sigma^2}\eta(t). \tag{31}$$

Comparing with (23), one finds

$$C(w) = w^2 + S\sigma^2. \tag{32}$$

The solution of the corresponding Fokker-Planck equation is

$$\begin{aligned}
P(w) &\propto \frac{1}{w^2 + S\sigma^2}\exp\left[-\frac{2S}{\lambda}\int\frac{(a/2)w + bw^3}{w^2 + S\sigma^2}dw\right] \\
&= (w^2 + S\sigma^2)^{-1}\exp\left[-\frac{2Sb}{\lambda}\int\frac{\left(\frac{a}{2b} - S\sigma^2\right)w + w^3 + S\sigma^2 w}{w^2 + S\sigma^2}dw\right] \\
&= (w^2 + S\sigma^2)^{-1}\exp\left[-\frac{2Sb}{\lambda}\int wdw - \frac{2Sb}{\lambda}\left(\frac{a}{2b} - S\sigma^2\right)\int\frac{w}{w^2 + S\sigma^2}dw\right] \\
&= (w^2 + S\sigma^2)^{-1}\exp\left[-\frac{Sb}{\lambda}w^2 - \left(\frac{Sa}{2\lambda} - \frac{S^2\sigma^2 b}{\lambda}\right)\ln\left(w^2 + S\sigma^2\right)\right].
\end{aligned}$$

Further simplification yields

$$P(w) \propto (w^2 + S\sigma^2)^{-1-\frac{Sa}{2\lambda}+\frac{S^2 b\sigma^2}{\lambda}}\exp\left[-\frac{Sb}{\lambda}w^2\right]. \tag{33}$$

The function $P(w)$ is finite everywhere and decays exponentially to 0 when $w \to \infty$, regardless of the values of the parameters a, b, σ, S. This indicates that $\int_{-\infty}^{\infty} dw(w^2 + S\sigma^2)^{-1-\frac{Sa}{2\lambda}+\frac{S^2 b\sigma^2}{\lambda}}\exp\left[-\frac{Sb}{\lambda}w^2\right]$ has a well-defined value. As a consequence, there is no concentration of measure. Thus, $w(t)$ can be fall into any interval with finite probability after big enough t, indicating that $w(t)$ can be arbitrarily far away from 0. In practice, this means that $w(t)$ escapes from the saddle point at $w = 0$ regardless of the value of the parameters.

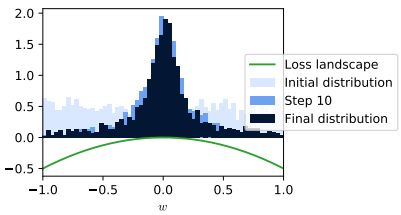

Figure 8: Stationary distribution of w with additive noise of $\sigma = 0.1$ with quadratic loss landscape. All the other parameters are identical to those in Fig.1. This distribution has finite width.

The effect of additive noise is better understood when $b = 0$. In this case, which corresponds to the case shown in Fig.1, w neither concentrates at 0, nor escapes to $\pm\infty$. Instead, it stays near 0 without

converging to it. The stationary distribution of w when $b = 0$, obtained by numerical simulation, is shown in Fig.8. The other settings are identical to the ones in Fig.1. The stationary distribution exists, but w does not concentrate at $w = 0$ in the stationary distribution. First of all, this figure shows that the additive noise helps SGD to escape saddle point. If the designed landscape is a part of a realistic landscape, having a broader distribution means having more chance of being attracted by another minimum. However, this figure also shows that w would stay in the neighborhood of $w = 0$ until the noise is big enough. Thus, having additive noise and long enough training time does not guarantee the efficiency of escape.

In the language of Bayesian inference, $\ln[P(w)]$ is the likelihood of w. $\hat{w}$, the most probable value of w, is defined by the relation

$$\left.\frac{d\ln[P(w)]}{dw}\right|_{w=\hat{w}} = 0. \tag{34}$$

which reads

$$\begin{aligned}
0 &= \left\{\frac{d}{dw}\left[\left(-1-\frac{Sa}{2\lambda}+\frac{S^2b\sigma^2}{\lambda}\right)\ln[(w^2+S\sigma^2)]-\frac{Sb}{\lambda}w^2\right]\right\}\Bigg|_{w=\hat{w}} \\
&= \left\{2w\frac{-1-\frac{Sa}{2\lambda}+\frac{S^2b\sigma^2}{\lambda}}{w^2+S\sigma^2}-\frac{2Sb}{\lambda}w\right\}\Bigg|_{w=\hat{w}} \\
&= \left\{w\left[-1-\frac{Sa}{2\lambda}+\frac{S^2b\sigma^2}{\lambda}-\frac{Sb}{\lambda}(w^2+S\sigma^2)\right]\right\}\Bigg|_{w=\hat{w}} \\
&= \hat{w}\left(\frac{\lambda}{Sb}+\frac{a}{2b}+\hat{w}^2\right).
\end{aligned}$$

The likelihood maximizer is

$$\hat{w} = \begin{cases} 0, & a > -\frac{2\lambda}{S}; \\ \pm\sqrt{-\frac{\lambda}{Sb}-\frac{a}{2b}}, & a < -\frac{2\lambda}{S}. \end{cases} \tag{35}$$

When $a > -\frac{2\lambda}{S}$, the maximum likelihood parameter is $w = 0$. When $a < -\frac{2\lambda}{S}$, $w = \pm\sqrt{-\frac{\lambda}{Sb}-\frac{a}{2b}}$ equally likely. This resembles the phase transition in statistical physics and we call $-\frac{2\lambda}{S}$ the critical value of a, or, the "critical a". For the energy landscape, when $a > 0$ there is only one global minimum at $w = 0$, and when $a < 0$ there are two global minima at $w = \pm\sqrt{-\frac{a}{2b}}$. Comparing with the maximum likelihood solutions, we see that the likelihood maximizer given by SGD is in fact a biased estimator of the underlying minima.

There is a bias term introduced by the SGD noise in both the critical a and the value of the most probable w. This term vanishes when $S \to \infty$, i.e. when the SGD noise vanishes. This indicates that, when the noise is state-dependent, there is no reason to expect SGD to be an unbiased or consistent estimator of the minimizer, as some works assume.

B.3 4-TH ORDER POTENTIAL WITH MULTIPLICATIVE NOISE

In this subsection, we study a 4-th order loss landscape in SGD dynamics without additive noise. The average loss function with 4th-order term is

$$L(w) = \frac{aw^2}{4}+\frac{bw^4}{4}; \tag{36}$$

the SGD update rule (i.e., the equation of motion) in this case is

$$dw(t) = -\lambda[(a/2)w(t)+bw^3(t)]dt+\lambda\frac{1}{\sqrt{S}}\eta(t)w(t), \tag{37}$$

where the definition of η is identical to the previous example. The solution of corresponding stationary Fokker-Planck equation is

$$P(w) \propto w^{-2-\frac{Sa}{\lambda}}\exp\left[-\frac{Sb}{\lambda}w^2\right]. \tag{38}$$

When

$$-\frac{Sa}{2\lambda} < 1, \tag{39}$$

$P(w)$ diverges at $w = 0$. The normalization factor, i.e. integral $\int_{-\infty}^{+\infty} dw w^{-2-\frac{Sa}{\lambda}} \exp\left[-\frac{Sb}{\lambda}w^2\right]$, also diverges due to the divergence at $w = 0$. This solution could be seen as a limit of (33) when the additive noise vanishes. As the strength of the additive noise $\sigma \to 0$, the normalization factor keep growing while the distribution remains normalized. For $w \neq 0$, the function $w^{-2-\frac{Sa}{\lambda}} \exp\left[-\frac{Sb}{\lambda}w^2\right]$ is always finite. Thus, in the limit $\sigma \to 0$, $P(w)$ approaches 0 everywhere except for at $w = 0$. It is straight forward to check that the normalization factor diverges slower that $P(0)$ as $\sigma \to 0$. As a consequence, when $\sigma = 0$, $P(w)$ diverges at $w = 0$ and $P(w)$ becomes a Dirac delta function. This corroborates with the result in Sec. 5.3 that a small a leads to a concentration of measure towards the local maximum.

Compared with (33), one finds that the additive noise prevents the concentration of measure. Thus, in practice, the additive noise helps SGD escape the local minimum or saddle point. However, the existence of additive noise does not change the critical a and the position of the peaks.

B.4 Quadratic Potential

At last, we consider the continuous approximation of the model treated in the Sec. 5.1. Let the loss function be

$$\hat{L} = \frac{x}{2}w^2, \tag{40}$$

and we have

$$\begin{cases} \mathbb{E}_x[\hat{g}_t] = \frac{a}{2}w_t \\ C(w_t) = \frac{1}{S}\mathbb{E}\left[\nabla\ell\nabla\ell^{\mathrm{T}}\right] - \frac{1}{S}\nabla; \quad L(\mathbf{w}_t)\nabla L(\mathbf{w}_t)^{\mathrm{T}} = \frac{1}{S}\mathbb{E}[x^2]w_t^2. \end{cases} \tag{41}$$

The SGD update rule (i.e., the equation of motion) in 1d is

$$\Delta w(t) = -\lambda\frac{\partial}{\partial w}\hat{L} \tag{42}$$

$$= -\lambda x w(t). \tag{43}$$

With the continuous approximation, the equation of motion becomes

$$dw(t) = -\lambda(a/2)w(t)dt + \lambda\frac{1}{\sqrt{S}}w(t)\eta(t), \tag{44}$$

where $\eta(t) \sim \mathcal{N}(0, \sqrt{dt})$. By comparing with the 1d Fokker-Planck equation,

$$\begin{cases} L(w) = \frac{a}{4}w^2; \\ B(w) = w. \end{cases} \tag{45}$$

The stationary distribution of w is

$$P(w) \propto \frac{1}{w^2}\exp\left[-\frac{S}{\lambda}\int dw\frac{a}{w}\right] \propto w^{-2-\frac{Sa}{\lambda}}. \tag{46}$$

The only difference between the solution in this case and the one in the 4th-order-loss case is the exponential factor, showing that the 4th-order potential does nothing but keeping $w(t)$ bounded. The function $w^{-2-\frac{Sa}{\lambda}}$ diverges at $w = 0$ or $w = \pm\infty$ depending on the value of a. However, the stationary distribution, being a limit of (33), is well defined. In the case that it diverges at $w = 0$, $P(w)$ becomes a delta function. On the contrary, there are two peaks infinitely far away from 0 when $w^{-2-\frac{Sa}{\lambda}}$ diverges at $w = \pm\infty$. Being able to escape requires that the probability measure of w does not concentrate at 0, corresponding to

$$-\frac{Sa}{2\lambda} > 1. \tag{47}$$

This straight line agrees approximately with the boundary of the lower branch of the phase diagram in Fig. 9. This condition of escaping is worth interpretation. Note that a is the local curvature and reflects the strength of the gradient signal. In contrast, the strength of the SGD noise is proportional

to λ/S.[2] The term $Sa/2\lambda$ is thus the signal to noise ratio of this learning problem, and the escaping condition is exactly when the signal becomes larger than the noise. This analysis, therefore, pinpoints the cause of the convergence to saddle points to be the dominance of the SGD noise over the gradient.

Also, this example raises an interesting question regarding the mechanism of SGD that causes a convergence to the local maximum. From the perspective of the types of convergence, the SGD mechanism behind proposition 1 may be different from that of proposition 2. However, from the perspective of continuous-time analysis, there is really the same mechanism that is governing the SGD dynamics behind proposition 1 and 2 (namely, the fact that the noise has dominated the gradient).

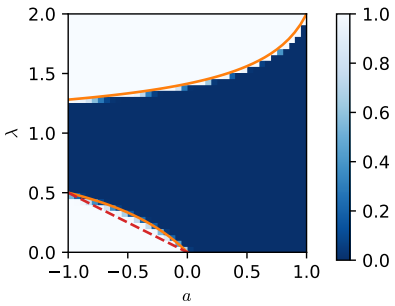

Figure 9: Escape probability as a function of a and λ. The parameters space is divided into an absorbing phase where w is attracted to the local maximum (in **dark blue**) and an active phase where w successfully escapes the two central bins (in white). The orange line denotes the analytical convergence bound on λ as a function of a obtained in the discrete-time calculation, while the red dashed line denotes the same bound obtained by the continuous-time calculation. Here the batch size S is set to 1. The result given by the continuous-time process agrees well with the discrete-time process in the small λ small a limit.

[2] See Ziyin et al. (2022b), for example.

C DELAYED PROOFS

C.1 TWO FREQUENTLY USED LEMMAS

We first prove the following lemma regarding the limiting distribution of $\ln|w_t|$.

Lemma 1. *Let the loss function be $\hat{L}(w) = \frac{1}{2}xw^2$, $x \sim p(x)$ such that $\mathrm{Var}[x] = \sigma^2$ and $\mathbb{E}_x[x] = a < 0$ and that $p(x)$ is continuous in a δ-neighborhood of $x = 1$, and $w_0 \neq 0$. Then, for w_t generated by SGD after t time steps,*

$$\frac{1}{\sqrt{t}}(\ln|w_t/w_0| - \mu) \to_d \mathcal{N}(0, s^2) \tag{48}$$

where $\mu = \mathbb{E}_x[\ln|1-\lambda x|]$ and $s^2 = \mathrm{Var}[\ln|1-\lambda x|]$, where $\lambda > 0$ is the learning rate.

Proof. After t steps of SGD, we have

$$w_t = \prod_{i=1}^{t}[1-\lambda x_i]w_0. \tag{49}$$

This leads to

$$\ln\left|\frac{w_t}{w_0}\right| = \sum_{i=1}^{t}\ln|1-\lambda x_i|. \tag{50}$$

Now, since $p(1-x)$ is continuous in the neighborhood of 1 by assumption, we can apply Proposition 6 (appendix) to show that the first and second moment of $\ln|1-\lambda x|$ exists, whereby we can apply the central limit theorem to obtain

$$\frac{1}{\sqrt{t}}\left(\ln\left|\frac{w_t}{w_0}\right| - t\mathbb{E}[\ln|1-\lambda x|]\right) \to_d \mathcal{N}(0, s^2), \tag{51}$$

where $s^2 = \mathrm{Var}[\ln|1-\lambda x|]$. This proves the lemma. □

Proposition 6. *$\mathbb{E}_{x\sim p(x)}[\ln^2|x|]$ is finite if the second moment of $p(x)$ exists and if $p(x)$ is continuous in the neighborhood of $x = 0$.*

Proof. Since $p(x)$ is continuous in the neighborhood of $x = 0$, then there exists $\delta > 0$ such that for all $|\epsilon| < \delta$, $|p(\delta) - p(x)| \leq c$ for some $c > 0$. This means that we can divide the integral over $p(x)$ into three regions:

$$\mathbb{E}_{x\sim p(x)}[\ln|x|^2] = \int_{-\infty}^{\infty} dx p(x)\ln^2|x| \tag{52}$$

$$= \int_{-\infty}^{-\delta} dx p(x)\ln^2|x| + \int_{-\delta}^{\delta} dx p(x)\ln^2|x| + \int_{\delta}^{\infty} dx p(x)\ln^2|x|. \tag{53}$$

The second term can be bounded as

$$\int_{-\delta}^{\delta} dx p(x)\ln^2|x| \leq [p(0)+c]\int_{-\delta}^{\delta} dx \ln^2|x| \tag{54}$$

$$= 2[p(0)+c]\int_{0}^{\delta} dx \ln^2 x \tag{55}$$

$$= 2[p(0)+c]x(\ln^2 x - 2\ln x + 2)\Big|_0^{\delta} \tag{56}$$

$$= 2[p(0)+c]\delta(\ln^2\delta - 2\ln\delta + 2) < \infty. \tag{57}$$

The first and the third term can also be bounded. Since $\ln|x|$ is a convex function in the regions $x \geq \delta$ and $x \leq \delta$ respectively. We can find linear functions $ax+b$ of x such that $ax+b \geq \ln x$ for $x \geq \delta$. This leads to

$$\int_{\delta}^{\infty} dx\, p(x)\ln^2|x| \leq \int_{\delta}^{\infty} dx\, p(x)(ax+b)^2 \tag{58}$$

$$\leq \int_{-\infty}^{\infty} dx\, p(x)(ax+b)^2 \tag{59}$$

$$= \int_{-\infty}^{\infty} dx p(x)(a^2x^2 + 2abx + b^2) \tag{60}$$

$$= a^2\mu_2 + 2ab\mu_1 + b^2, \tag{61}$$

where we have used the notation $\mathbb{E}[x^2] = \mu_2$ and $\mathbb{E}[x^1] = \mu$, which by assumption is finite. The case for $x \leq -\delta$ is completely symmetric and can also be bounded by $a^2\mu_2 + 2ab\mu_1 + b^2$. Therefore, we have shown that

$$\mathbb{E}_{x\sim p(x)}[\ln|x|^2] \leq 2(a^2\mu_2 + 2ab\mu_1 + b^2) + 2(p(0) + c)\delta(\ln^2\delta - 2\ln\delta + 2) \leq \infty. \tag{62}$$

This proves the proposition. □

Remark. *Note that, the continuity in the neighborhood of* 0 *is not a necessary condition. For example, one can also prove that* $\mathbb{E}_x[\ln^2|x|]$ *is finite if* $p(x)$ *is bounded and its second moment exists.*

C.2 Proof of Proposition 2

Proof. The case when $w_0 = 0$ is trivially true, we therefore consider the case $w_0 \neq 0$. By Lemma 1, we have

$$\frac{1}{\sqrt{t}} z_t \to_d \mathcal{N}(0, s^2) \tag{63}$$

where we have defined $z_t = \frac{1}{\sqrt{t}}(\ln\left|\frac{w_t}{w_0}\right| - t\mu)$, $\mu = \mathbb{E}[\ln|1-\lambda x|]$ and $s^2 = \mathrm{Var}[\ln|1-\lambda x|]$. This means that

$$\lim_{t\to\infty} p\left(\frac{1}{\sqrt{t}}\ln\left|\frac{w_t}{w_0}\right|\right) = \lim_{t\to\infty}\frac{1}{\sqrt{2\pi s^2}}\exp\left[-\frac{1}{2ts^2}\left(\ln\left|\frac{w_t}{w_0}\right| - t\mu\right)\right]. \tag{64}$$

By definition, $|w_t| = |w_0|e^{\sqrt{t}z+t\mu}$. Therefore, we have

$$\lim_{t\to\infty} P(|w_t| > \epsilon) = \lim_{t\to\infty} P\left(|w_0|e^{\sqrt{t}z+t\mu} > \epsilon\right) = \begin{cases} 0 & \text{if } \mu < 0; \\ 1 & \text{if } \mu > 0, \end{cases} \tag{65}$$

for all $\epsilon > 0$. In other words, since μ and σ are t-independent, in the infinite t limit, the sign of μ becomes crucial. When $\mu > 0$, the limiting distribution diverges to infinity; when $\mu < 0$, $|w_t|$ converges to 0 in probability. This finishes the proof. □

C.3 Proof of Corollary 1

Proof. By the definition of $p(x)$,

$$\mu = \mathbb{E}[\ln|1-\lambda x|] \tag{66}$$

$$= \frac{1}{2}\ln|1-\lambda| + \frac{1}{2}\ln|1-\lambda(a-1)| \tag{67}$$

$$= \frac{1}{2}\ln|(1-\lambda)(1-\lambda(a-1))|, \tag{68}$$

while s^2 is indeed constant in time. Therefore, the asymptotic distribution of w_t is solely dependent on the sign of μ. The above equation implies that $\mu \geq 0$ when

$$|(1-\lambda)[1-\lambda(a-1)]| > 1. \tag{69}$$

When $\lambda \leq 1$, the above equation is solved by

$$\lambda < \frac{a}{a-1}. \tag{70}$$

When $\lambda > 1$, the above equation solves to

$$\lambda \geq \frac{a - \sqrt{a^2 - 8a + 8}}{2(a-1)}, \tag{71}$$

only when the learning rate satisfies the above two conditions can it escape from the local maximum. Conversely, SGD cannot escape the local minimum when

$$\frac{a}{a-1} \leq \lambda \leq \frac{a - \sqrt{a^2 - 8a + 8}}{2(a-1)}. \tag{72}$$

We are done. □

C.4 Proof of Proposition 3

Proof. By Lemma 1, we have that

$$\frac{1}{t}\mathbb{E}\left[\ln\left|\frac{w_t}{w_0}\right|\right] = \mu = \frac{1}{2}\ln\{(1-\lambda)[1-\lambda(a-1)]\}, \tag{73}$$

where the second equality follows from the assumption that $\lambda < 1$. We differentiate with respect to λ to find the critical escape rate:

$$\lambda^* = \frac{a}{2(a-1)}. \tag{74}$$

Since μ is convex in λ, it follows that this critical escape rate is the maximum escape rate. Plugging this into μ, we obtain that

$$\gamma^* = \mu(\lambda^*) = \frac{1}{2}\ln\frac{(2-a)^2}{4(1-a)} = \ln\frac{2-a}{2\sqrt{1-a}} \leq \epsilon, \tag{75}$$

i.e., the optimal escape rate can be made smaller than any ϵ if we set

$$|a| \leq 2\left|-e^{\epsilon}\sqrt{e^{2\epsilon}-1} - e^{2\epsilon} + 1\right|. \tag{76}$$

This finishes the proof. □

C.5 Proof of Proposition 4

Proof. It suffices to show that w_1 converges to 0 with probability 1 because, if this is the case, the only possible local minimum to converge to is $(w_1, w_2) = (0, \pm\sqrt{\frac{1+a}{2}})$.

The SGD dynamics is

$$\begin{cases}\Delta w_{1,t} = -\lambda(-2w_{1,t} + 4w_{1,t}^3 + 12w_{2,t}^2 w_{1,t} + x_t b w_{1,t});\\ \Delta w_{2,t} = -\lambda(-2w_{2,t} + 4w_{2,t}^3 + 12w_{1,t}^2 w_{2,t} - 2a w_{2,t}).\end{cases} \tag{77}$$

We focus on the dynamics of w_1. By the definition of the noise x, we have that

$$w_{1,t+1} = \begin{cases} w_{1,t}[1+\lambda(2-b) - 4\lambda w_{1,t}^2 - 12\lambda w_{2,t}^2] & \text{with probability } 1/2;\\ w_{1,t}[1+\lambda(2+b) - 4\lambda w_{1,t}^2 - 12\lambda w_{2,t}^2] & \text{with probability } 1/2.\end{cases} \tag{78}$$

Equivalently,

$$\left|\frac{w_{1,t+1}}{w_{1,t}}\right| = \begin{cases} \left|1+\lambda(2-b) - 4\lambda w_{1,t}^2 - 12\lambda w_{2,t}^2\right| & \text{with probability } 1/2;\\ \left|1+\lambda(2+b) - 4\lambda w_{1,t}^2 - 12\lambda w_{2,t}^2\right| & \text{with probability } 1/2.\end{cases} \tag{79}$$

Since $0 \leq 4\lambda w_{1,t}^2 + 12\lambda w_{2,t}^2 \leq 16\lambda$, we can define a new random variable r_t:

$$r_t := r(x_t) := \begin{cases}\max(|1+\lambda(2-b)|, |1+\lambda(2-b)-16\lambda|) & \text{if } x_t \geq 0;\\ \max(|1+\lambda(2+b)|, |1+\lambda(2+b)-16\lambda|) & \text{if } x_t \leq 0.\end{cases} \tag{80}$$

By construction, $r_t \geq |w_{1,t+1}/w_{1,t}|$ for all values of x_t. This implies that $|w_{1,t}/w_{1,0}| \leq \prod_{i=1}^t r_t$, and so,

$$P(|w_{1,t}/w_{1,0}| > \epsilon) \leq P\left(\left|\prod_{i=1}^t r_t\right| > \epsilon\right), \tag{81}$$

i.e., if $\prod_{i=1}^t r_t$ converges to 0 with probability 1, $w_{1,t}$ must also converge to zero with probability 1.

We let $b = \frac{1}{\lambda} - 6$ (note that this is the value of b such that r_t is minimized for both cases), and we obtain that

$$r_t = \begin{cases} 8\lambda & \text{if } x_t \geq 0;\\ 2\max(|1-2\lambda|, |1-10\lambda|) & \text{if } x_t \leq 0.\end{cases} \tag{82}$$

The rest of the proof follows from applying the central limit theorem to $\frac{1}{\sqrt{t}}\sum_{i=1}^t \ln r_t$ as in Lemma 1. The result is that $\prod_{i=1}^t r_i$ converges to 0 with probabiliy 1 if

$$\mu := \mathbb{E}[\ln r_t] = \frac{1}{2}\ln[8\lambda\max(|1-2\lambda|, |1-10\lambda|)] < 0. \tag{83}$$

The above inequality solves to

$$\lambda \leq \frac{1}{20}(1+\sqrt{6}) \tag{84}$$

which is of order $O(1)$ as stated in the theorem statement. This proves the proposition. □

C.6 PROOF OF PROPOSITION 5

Proof. First we note that, since $|w_t| \leq 1$

$$|\hat{g}_t| = |x_t w_t| \leq 1 + a. \tag{85}$$

By the definition of the AMSGrad algorithm, we have that

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\hat{g}_t^2; \tag{86}$$
$$\hat{v}_t = \max(\hat{v}_{t-1}, v_t). \tag{87}$$

Since $v_0 = 0$ and $|\hat{g}_t| = |x_t w_t| \leq 1 + a$, we have that

$$v_t \leq 1 + a \tag{88}$$

for all t, and so

$$\hat{v}_t \leq \max_t \hat{v}_t = \max_t v_t \leq 1 + a. \tag{89}$$

Meanwhile, since $\hat{v}_t$ is a monotonically increasing series and is upper bounded by $1 + a$, it must converge to a constant $0 < c \leq 1 + a$.

Now, as before, we want to upper bound the random variable

$$\frac{1}{\sqrt{t}}\left(\ln\left|\frac{w_{t+1}}{w_t}\right| - \mu\right) = \frac{1}{\sqrt{t}}\sum_{i=1}^{t}\left(\ln\left|1 - \frac{\lambda}{\sqrt{\hat{v}_t}}x_t\right| - \mu\right), \tag{90}$$

where $\mu = \mathbb{E}[\ln\left|\frac{w_{t+1}}{w_t}\right|]$. Since $\hat{v}_t$ converges to c, there must exists a positive integer $N(\epsilon)$ such that for any $\epsilon > 0$, $\hat{v}_N \leq c - \epsilon$. This allows us to divide the sum to two terms:

$$\frac{1}{\sqrt{t}}\left(\ln\left|\frac{w_{t+1}}{w_t}\right| - \mu\right) = \frac{1}{\sqrt{t}}\sum_{i=1}^{t}\left(\ln\left|1 - \frac{\lambda}{\sqrt{\hat{v}_t}}x_t\right| - \mu\right) \tag{91}$$

$$= \frac{1}{\sqrt{t}}\sum_{i=1}^{N}\left(\ln\left|1 - \frac{\lambda}{\sqrt{\hat{v}_t}}x_t\right| - \mu\right) + \frac{1}{\sqrt{t}}\sum_{i=N+1}^{t}\left(\ln\left|1 - \frac{\lambda}{\sqrt{c-k_t}}x_t\right| - \mu\right), \tag{92}$$

where we introduced $0 \leq k_t \leq \epsilon$. But the first term is of order $O(1/\sqrt{t})$ and converges to 0, and so for sufficiently large t, the first term is also smaller than arbitrary ϵ. This means that

$$\frac{1}{\sqrt{t}}\left(\ln\left|\frac{w_{t+1}}{w_t}\right| - \mu\right) \leq \frac{1}{\sqrt{t}}\sum_{i=1}^{t}\left(\ln\left|1 - \frac{\lambda}{\sqrt{c-k_t}}x_t\right| - \mu\right) + \epsilon, \tag{93}$$

We can now consider the random variable

$$\left|1 - \frac{\lambda}{\sqrt{c-k_t}}x_t\right| = \begin{cases}\left|1 - \frac{\lambda}{\sqrt{\hat{v}_t}}\right| & \text{with probability } \frac{1}{2};\\ 1 + \frac{\lambda}{\sqrt{\hat{v}_t}}(1-a) & \text{with probability } \frac{1}{2}.\end{cases} \tag{94}$$

We now define a new random variable

$$r_t(x_t) := \begin{cases} m & \text{if } x_t = 1;\\ 1 + \frac{\lambda}{\sqrt{c-\epsilon}}(1-a) & \text{if } x_t = -1 + a.\end{cases} \tag{95}$$

where we defined the constant $m := \max(|1 - \lambda/\sqrt{c}|, |1 - \lambda/\sqrt{c-\epsilon}|)$. One can easily check that $r_t \geq \left|1 - \frac{\lambda}{\sqrt{c-k_t}}x_t\right|$. This means that

$$P\left(\left|\frac{w_{t+1}}{w_0}\right| > \alpha\right) \leq P\left(\prod_{i=0}^{t} r_i > \alpha + O(1/\sqrt{t})\right), \tag{96}$$

i.e., if r_t converges to 0 in probability, $|\frac{w_{t+1}}{w_0}|$ must also converge to 0 in probability. Proceeding as in the proof of Proposition 2. One finds that the condition for r_t to converge in probability to 0 is

$$\ln\left|m\left[1 + \frac{\lambda}{\sqrt{c-\epsilon}}(1-a)\right]\right| < 0, \tag{97}$$

which is equivalent to

$$\left| m\left[1 + \frac{\lambda}{\sqrt{c}}(1-a)\right]\right| < 1. \tag{98}$$

Since ϵ is arbitrary, we let $\epsilon \to 0$ and obtain

$$\left|\left(1 - \frac{\lambda}{\sqrt{c}}\right)\left[1 + \frac{\lambda}{\sqrt{c}}(1-a)\right]\right| \leq 1. \tag{99}$$

Denote $\lambda/\sqrt{c}$ as λ', this condition solves to

$$\frac{a}{a-1} \leq \lambda' \leq \frac{a - \sqrt{a^2 - 8a + 8}}{2(a-1)}. \tag{100}$$

Setting a such that the above condition is met, AMSGrad will converge to 0 in probability. For $\lambda < 1$, This completes the proof. □

Publication 5: Exact solutions of a deep linear network

Exact Solutions of a Deep Linear Network

Liu Ziyin[1], **Botao Li**[2], **Xiangming Meng**[3]
[1]Department of Physics, The University of Tokyo
[2]Laboratoire de Physique de l'Ecole normale supérieure, ENS, Université PSL, CNRS, Sorbonne Université, Université de Paris Cité, Paris, France
[3]Institute for Physics of Intelligence, Graduate School of Science, The University of Tokyo

Abstract

This work finds the analytical expression of the global minima of a deep linear network with weight decay and stochastic neurons, a fundamental model for understanding the landscape of neural networks. Our result implies that zero is a special point in deep neural network architecture. We show that weight decay strongly interacts with the model architecture and can create bad minima at zero in a network with more than 1 hidden layer, qualitatively different from a network with only 1 hidden layer. Practically, our result implies that common deep learning initialization methods are insufficient to ease the optimization of neural networks in general.

1 Introduction

Applications of neural networks have achieved great success in various fields. One central open question is why neural networks, being nonlinear and containing many saddle points and local minima, can sometimes be optimized easily (Choromanska et al., 2015a) while becoming difficult and requiring many tricks to train in some other scenarios (Glorot and Bengio, 2010; Gotmare et al., 2018). One established approach is to study the landscape of deep linear nets (Choromanska et al., 2015b), which are believed to approximate the landscape of a nonlinear net well. A series of works proved the famous results that for a deep linear net, all local minima are global (Kawaguchi, 2016; Lu and Kawaguchi, 2017; Laurent and Brecht, 2018), which is regarded to have successfully explained why deep neural networks are so easy to train because it implies that initialization in any attractive basin can reach the global minimum without much effort (Kawaguchi, 2016). However, the theoretical problem of when and why neural networks can be hard to train is understudied.

In this work, we theoretically study a deep linear net with weight decay and stochastic neurons, whose loss function takes the following form in general:

$$\underbrace{\mathbb{E}_x \mathbb{E}_{\epsilon^{(1)},\epsilon^{(2)},...,\epsilon^{(D)}} \left(\sum_{i,i_1,i_2,...,i_D}^{d,d_1,d_2,...d_D} U_{i_D} \epsilon^{(D)}_{i_D} ... \epsilon^{(2)}_{i_2} W^{(2)}_{i_2 i_1} \epsilon^{(1)}_{i_1} W^{(1)}_{i_1 i} x_i - y \right)^2}_{L_0} + \underbrace{\gamma_u ||U||_2^2 + \sum_{i=1}^{D} \gamma_i ||W^{(i)}||_F^2}_{L_2 \ reg.}, \tag{1}$$

where $\mathbb{E}_x$ denotes the expectation over the training set, U and $W^{(i)}$ are the model parameters, D is the depth of the network,[1] ϵ is the noise in the hidden layer (e.g., due to dropout), d_i is the width of the i-th layer, and γ is the strength of the weight decay. Previous works have studied special cases of this loss function. For example, Kawaguchi (2016) and Lu and Kawaguchi (2017) study the landscape of L_0 when ϵ is a constant (namely, when there is no noise). Mehta et al. (2021) studies

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

[1]In this work, we use "depth" to refer to the number of hidden layers. For example, a linear regressor has depth 0.

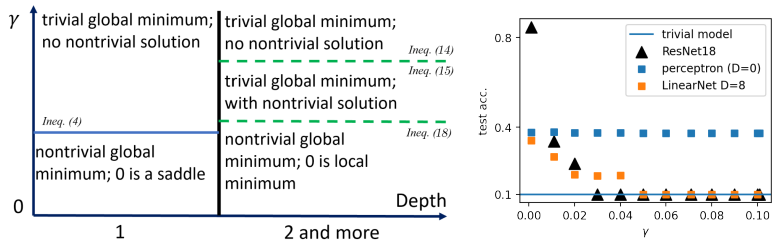

Figure 1: **Left**: A summary of the network landscape that is implied by the main results of this work when one increases the weight decay strength γ while fixing other terms. We show that the landscape of a depth-1 net can be precisely divided into two regimes, while, for $D \geq 2$, there exists at least three regimes. The solid blue line indicates that the division of the regimes is precisely understood. The dashed lines indicate that the conditions we found are not tight and may be improved in the future. **Right**: ResNet18 on CIFAR10. The performance of a linear regressor never drops to that of a trivial model, whereas the performance of ResNet18 drops to the level of a trivial model, like a deep linear net with similar depth.

L_0 with (a more complicated type of) weight decay but without stochasticity and proved that all the stationary points are isolated. Another line of works studies L_0 when the noise is caused by dropout (Mianjy and Arora, 2019; Cavazza et al., 2018). Our setting is more general than the previous works in two respects. First, apart from the mean square error (MSE) loss L_0, an L_2 regularization term (weight decay) with arbitrary strength is included; second, the noise ϵ is arbitrary. Thus, our setting is arguably closer to the actual deep learning practice, where the injection of noises to latent layers is common, and the use of weight decay is virtually ubiquitous (Krogh and Hertz, 1992; Loshchilov and Hutter, 2017). One major limitation of our work is that we assume the label y to be 1-dimensional, and it can be an important future problem to prove whether an exact solution exists or not when y is high-dimensional.

Our foremost contribution is to prove that all the global minimum of an arbitrarily deep and wide linear net takes a simple analytical form. In other words, we identify in closed form the global minima of Eq. (1) up to a single scalar, whose analytical expression does not exist in general. We then show that it has nontrivial properties that can explain many phenomena in deep learning. In particular, the implications of our result include (but are not limited to):

1. Weight decay makes the landscape of neural nets more complicated;
   - we show that bad minima[2] emerge as weight decay is applied, whereas there is no bad minimum when there is no weight decay. This highlights the need to escape bad local minima in deep learning with weight decay.
2. Deeper nets are harder to optimize than shallower ones;
   - we show that a $D \geq 2$ linear net contains a bad minimum at zero, whereas a $D = 1$ net does not. This partially explains why deep networks are much harder to optimize than shallower ones in deep learning practice.
3. Depending on the task, the common initialization methods (such as the Kaiming init.) can initialize a deep model in the basin of attraction of the bad minimum at zero;
   - common initialization methods initialize the models at a radius of roughly $1/\sqrt{width}$ around the origin; however, we show that the width of the bad minimum is task-dependent and can be larger than the initialization radius for tasks with a small margin ($\|\mathbb{E}[xy]\|$);
4. Thus, the use of (effective) weight decay is a major cause of various types of collapses in deep learning (for example, see Figure 1).

Organization: In the next section, we discuss the related works. In Section 3, we derive the exact solution for a two-layer net. Section 4 extends the result to an arbitrary depth. In Section 5, we study and discuss the relevance of our results to many commonly encountered problems in deep learning. The last section concludes the work and discusses unresolved open problems. All proofs are delayed to Section B. Moreover, additional theoretical results on the effect of including a bias term is considered in Section D.

[2]Unless otherwise specified, we use the word "bad minimum" to mean a local minimum that is not a global minimum.

Notation. For a matrix W, we use $W_{i:}$ to denote the i-th row vector of W. $\|Z\|$ denotes the L_2 norm if Z is a vector and the Frobenius norm if Z is a matrix. The notation $*$ signals an optimized quantity. Additionally, we use the superscript * and subscript $_*$ interchangeably, whichever leads to a simpler expression. For example, b_*^2 and $(b^*)^2$ denote the same quantity, while the former is "simpler."

2 Related Works

In many ways, linear networks have been used to help understand nonlinear networks. For example, even at depth 0, where the linear net is just a linear regressor, linear nets are shown to be relevant for understanding the generalization behavior of modern overparametrized networks (Hastie et al., 2019). Saxe et al. (2013) studies the training dynamics of a depth-1 network and uses it to understand the dynamics of learning of nonlinear networks. These networks are the same as a linear regression model in terms of expressivity. However, the loss landscape is highly complicated due to the existence of more than one layer, and linear nets are widely believed to approximate the loss landscape of a nonlinear net (Kawaguchi, 2016; Hardt and Ma, 2016; Laurent and Brecht, 2018). In particular, the landscape of linear nets has been studied as early as 1989 in Baldi and Hornik (1989), which proposed the well-known conjecture that all local minima of a deep linear net are global. This conjecture is first proved in Kawaguchi (2016), and extended to other loss functions and deeper depths in Lu and Kawaguchi (2017) and Laurent and Brecht (2018). Many relevant contemporary deep learning problems can be understood with deep linear models. For example, two-layer linear VAE models are used to understand the cause of the posterior collapse problem (Lucas et al., 2019; Wang and Ziyin, 2022). Deep linear nets are also used to understand the neural collapse problem in contrastive learning (Tian, 2022). We also provide more empirical evidence in Section 5.

3 Two-layer Linear Net

This section finds the global minima of a two-layer linear net. The data point is a d-dimensional vector $x \in \mathbb{R}^d$ drawn from an arbitrary distribution, and the labels are generated through an arbitrary function $y = y(x) \in \mathbb{R}$. For generality, we let different layers have different strengths of weight decay even though they often take the same value in practice. We want to minimize the following objective:

$$L_{d,d_1}(U,W) = \mathbb{E}_x \mathbb{E}_\epsilon \left(\sum_j^{d_1} U_j \epsilon_j \sum_i^d W_{ji} x_i - y \right)^2 + \gamma_w \|W\|^2 + \gamma_u \|U\|^2, \tag{2}$$

where d_1 is the width of the hidden layer and ϵ_i are independent random variables. $\gamma_w > 0$ and $\gamma_u > 0$ are the weight decay parameters. Here, we consider a general type of noise with $\mathbb{E}[\epsilon_i] = 1$ and $\mathbb{E}[\epsilon_i \epsilon_j] = \delta_{ij}\sigma^2 + 1$ where δ_{ij} is the Kronecker's delta, and $\sigma^2 > 0$.[3] For shorthand, we use the notation $A_0 := \mathbb{E}[xx^T]$, and the largest and the smallest eigenvalues of A_0 are denoted as $a_{\max}$ and $a_{\min}$ respectively. a_i denotes the i-th eigenvalue of A_0 viewed in any order. For now, it is sufficient for us to assume that the global minimum of Eq. (2) always exists. We will prove a more general result in Proposition 1, when we deal with multilayer nets.

3.1 Main Result

We first present two lemmas showing that the global minimum can only lie on a rather restrictive subspace of all possible parameter settings due to invariances in the objective.

Lemma 1. *At the global minimum of Eq. (2),* $U_j^2 = \frac{\gamma_w}{\gamma_u} \sum_i W_{ji}^2$ *for all* j.

Proof Sketch. We use the fact that the first term of Eq. (2) is invariant to a simultaneous rescaling of rows of the weight matrix to find the optimal rescaling, which implies the lemma statement. □

This lemma implies that for all j, $|U_j|$ must be proportional to the norm of its corresponding row vector in W. This lemma means that using weight decay makes all layers of a deep neural network balanced. This lemma has been referred to as the "weight balancing" condition in recent works (Tanaka et al., 2020), and, in some sense, is a unique and potentially essential feature of neural networks that encourages a sparse solution (Ziyin and Wang, 2022). The following lemma further shows that, at the global minimum, all elements of U must be equal.

[3] While we formally require γ and σ to nonzero, one can show that the solutions we provided remain global minimizers in the zero limit by applying Theorem 2 from Ziyin and Ueda (2022).

Lemma 2. *At the global minimum, for all i and j, we have*

$$\begin{cases} U_i^2 = U_j^2; \\ U_i W_{i:} = U_j W_{j:}. \end{cases} \tag{3}$$

Proof Sketch. We show that if the condition is not satisfied, then an "averaging" transformation will strictly decrease the objective. □

This lemma can be seen as a formalization of the intuition suggested in the original dropout paper (Srivastava et al., 2014). Namely, using dropout encourages the neurons to be independent of one another and results in an averaging effect. The second lemma imposes strong conditions on the solution of the problem, and the essence of this lemma is the reduction of the original problem to a lower dimension. We are now ready to prove our first main result.

Theorem 1. *The global minimum U_* and W_* of Eq. (2) is $U_* = 0$ and $W_* = 0$ if and only if*

$$||\mathbb{E}[xy]||^2 \leq \gamma_u \gamma_w. \tag{4}$$

When $||\mathbb{E}[xy]||^2 > \gamma_u \gamma_w$, the global minima are

$$\begin{cases} U_* = b\mathbf{r}; \\ W_* = \mathbf{r}\mathbb{E}[xy]^T b \left[b^2 \left(\sigma^2 + d_1 \right) A_0 + \gamma_w I \right]^{-1}, \end{cases} \tag{5}$$

where $\mathbf{r} = (\pm 1, ..., \pm 1)$ is an arbitrary vertex of a d_1-dimensional hypercube, and b satisfies:

$$\left\| \left[b^2 \left(\sigma^2 + d_1 \right) A_0 + \gamma_w I \right]^{-1} \mathbb{E}[xy] \right\|^2 = \frac{\gamma_u}{\gamma_w}. \tag{6}$$

Apparently, $b = 0$ is the trivial solution that has not learned any feature due to overregularization. Henceforth, we refer to this solution (and similar solutions for deeper nets) as the "trivial" solution. We now analyze the properties of the nontrivial solution b^* when it exists.

The condition for the solution to become nontrivial is interesting: $||\mathbb{E}[xy]||^2 \geq \gamma_u \gamma_w$. The term $||\mathbb{E}[xy]||$ can be seen as the effective strength of the signal, and $\gamma_u \gamma_w$ is the strength of regularization. This precise condition means that the learning of a two-layer can be divided into two qualitatively different regimes: an "overregularized regime" where the global minimum is trivial, and a "feature learning regime" where the global minimum involves actual learning. Lastly, note that our main result does not specify the exact value of b^*. This is because b^* must satisfy the condition in Eq. (6), which is equivalent to a high-order polynomial in b with coefficients being general functions of the eigenvalues of A_0, whose solutions are generally not analytical by Galois theory. One special case where an analytical formula exists for b is when $A_0 = \sigma_x^2 I$. See Section C for more discussion.

3.2 Bounding the General Solution

While the solution to b^* does not admit an analytical form for a general A_0, one can find meaningful lower and upper bounds to b^* such that we can perform an asymptotic analysis of b^*. At the global minimum, the following inequality holds:

$$\begin{aligned} || \left[b^2 \left(\sigma^2 + d_1 \right) a_{\max} I + \gamma_w I \right]^{-1} \mathbb{E}[xy]||^2 \leq ||[b^2 \left(\sigma^2 + d_1 \right) A_0 + \gamma_w I]^{-1} \mathbb{E}[xy]||^2 \\ \leq || \left[b^2 \left(\sigma^2 + d_1 \right) a_{\min} I + \gamma_w I \right]^{-1} \mathbb{E}[xy]||^2, \end{aligned} \tag{7}$$

where $a_{\min}$ and $a_{\max}$ are the smallest and largest eigenvalue of A_0, respectively. The middle term is equal to γ_u / γ_w by the global minimum condition in (33), and so, assuming $a_{\min} > 0$, this inequality is equivalent to the following inequality of b^*:

$$\frac{\sqrt{\frac{\gamma_w}{\gamma_u}} ||\mathbb{E}[xy]|| - \gamma_w}{(\sigma^2 + d_1) a_{\max}} \leq b_*^2 \leq \frac{\sqrt{\frac{\gamma_w}{\gamma_u}} ||\mathbb{E}[xy]|| - \gamma_w}{(\sigma^2 + d_1) a_{\min}}. \tag{8}$$

Namely, the general solution b^* should scale similarly to the homogeneous solution in Eq. (105) if we treat the eigenvalues of A_0 as constants.

4 Exact Solution for An Arbitrary-Depth Linear Net

This section extends our result to multiple layers. We first derive the analytical formula for the global minimum of a general arbitrary-depth model. We then show that the landscape for a deeper network is highly nontrivial.

4.1 General Solution

The loss function is

$$\mathbb{E}_x \mathbb{E}_{\epsilon^{(1)},\epsilon^{(2)},\ldots,\epsilon^{(D)}} \left(\sum_{i,i_1,i_2,\ldots,i_D}^{d,d_1,d_2,\ldots d_D} U_{i_D} \epsilon_{i_D}^{(D)} \ldots \epsilon_{i_2}^{(2)} W_{i_2 i_1}^{(2)} \epsilon_{i_1}^{(1)} W_{i_1 i}^{(1)} x_i - y \right)^2 + \gamma_u \|U\|^2 + \sum_{i=1}^{D} \gamma_i \|W^{(i)}\|^2, \tag{9}$$

where all the noises ϵ are independent, and for all i and j, $\mathbb{E}[\epsilon_j^{(i)}] = 1$ and $\mathbb{E}[(\epsilon_j^{(i)})^2] = \sigma_i^2 + 1 > 1$. We first show that for general D, the global minimum exists for this objective.

Proposition 1. *For $D \geq 1$ and strictly positive $\gamma_u,\ \gamma_1, \ldots, \gamma_D$, the global minimum for Eq.(9) exists.*

Note that the positivity of the regularization strength is crucial. If one of the γ_i is zero, the global minimum may not exist. The following theorem is our second main result.

Theorem 2. *Any global minimum of Eq. (9) is of the form*

$$\begin{cases} U = b_u \mathbf{r}_D; \\ W^{(i)} = b_i \mathbf{r}_i \mathbf{r}_{i-1}^T; \\ W^{(1)} = \mathbf{r}_1 \mathbb{E}[xy]^T (b_u \prod_{i=2}^{D} b_i) \mu \left[(b_u \prod_{i=2}^{D} b_i)^2 s^2 \left(\sigma^2 + d_1\right) A_0 + \gamma_w I \right]^{-1}, \end{cases} \tag{10}$$

where $\mu = \prod_{i=2}^{D} d_i$, $s^2 = \prod_{i=2}^{D} d_i(\sigma^2 + d_i)$, $b_u \geq 0$ and $b_i \geq 0$, and $\mathbf{r}_i = (\pm 1, \ldots, \pm 1)$ is an arbitrary vertex of a d_i-dimensional hypercube for all i. Furthermore, let $b_1 := \sqrt{\|W_{i:}\|^2/d}$ and $b_{D+1} := b_u$, b_i satisfies

$$\gamma_{k+1} d_{k+1} b_{k+1}^2 = \gamma_k d_{k-1} b_k^2. \tag{11}$$

Proof Sketch. We prove by induction on the depth D. The base case is proved in Theorem 1. We then show that for a general depth, the objective involves optimizing subproblems, one of which is a $D-1$ layer problem that follows by the induction assumption, and the other is a two-layer problem that has been solved in Theorem 1. Putting these two subproblems together, one obtains Eq. (10). □

Remark. *We deal with the technical case of having a bias term for each layer in Appendix D. For example, we will show that if one has preprocessed the data such that $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$, our main results remain precisely unchanged.*

The condition in Eq. (11) shows that the scaling factor b_i for all i is not independent of one another. This automatic balancing of the norm of all layers is a consequence of the rescaling invariance of the multilayer architecture and the use of weight decay. It is well-known that this rescaling invariance also exists in a neural network with the ReLU activation, and so this balancing condition is also directly relevant for ReLU networks.

Condition (11) implies that all the b_i can be written in terms of one of the b_i:

$$b_u \prod_{i=2}^{D} b_i = c_0 \mathrm{sgn}\left(b_u \prod_{i=2}^{D} b_i \right) |b_2^D| := c_0 \mathrm{sgn}\left(b_u \prod_{i=2}^{D} b_i \right) b^D \tag{12}$$

where $c_0 = \frac{(\gamma_2 d_2 d_1)^{D/2}}{\sqrt{\gamma_u \prod_{i=2}^{D} \gamma_i} \prod_{i=2}^{D} d_i \sqrt{d_1}}$ and $b \geq 0$. Consider the first layer ($i = 1$), Eq (11) shows that the global minimum must satisfy the following equation, which is equivalent to a high-order polynomial in b that does not have an analytical solution in general:

$$\left\| \mathbb{E}[xy]^T c_0 b^D \mu \left[c_0^2 b^{2D} s^2 \left(\sigma^2 + d_1\right) A_0 + \gamma_w I \right]^{-1} \right\|^2 = d_2 b^2. \tag{13}$$

Thus, this condition is an extension of the condition (6) for two-layer networks.

At this point, it pays to clearly define the word “solution," especially given that it has a special meaning in this work because it now becomes highly nontrivial to differentiate between the two types of solutions.

Definition 1. *We say that a non-negative real b is a solution if it satisfies Eq. (13). A solution is trivial if $b = 0$ and nontrivial otherwise.*

Namely, a global minimum must be a solution, but a solution is not necessarily a global minimum. We have seen that even in the two-layer case, the global minimum can be the trivial one when the strength of the signal is too weak or when the strength of regularization is too strong. It is thus natural to expect 0 to be the global minimum under a similar condition, and one is interested in whether the condition becomes stronger or weaker as the depth of the model is increased. However, it turns out this naive expectation is not true. In fact, when the depth of the model is larger than 2, the condition for the trivial global minimum becomes highly nontrivial.

The following proposition shows why the problem becomes more complicated. In particular, we have seen that in the case of a two-layer net, some elementary argument has helped us show that the trivial solution $b = 0$ is either a saddle or the global minimum. However, the proposition below shows that with $D \geq 2$, the landscape becomes more complicated in the sense that the trivial solution is always a local minimum, and it becomes difficult to compare the loss value of the trivial solution with the nontrivial solution because the value of b^* is unknown in general.

Proposition 2. *Let $D \geq 2$ in Eq. (9). Then, the solution $U = 0$, $W^{(D)} = 0$, ..., $W^{(1)} = 0$ is a local minimum with a diagonal positive-definite Hessian γI.*

Comparing the Hessian of $D \geq 2$ and $D = 1$, one notices a qualitative difference: for $D \geq 2$, the Hessian is always diagonal (at 0); for $D = 1$, in sharp contrast, the off-diagonal terms are nonzero in general, and it is these off-diagonal terms that can break the positive-definiteness of the Hessian. This offers a different perspective on why there is a qualitative difference between $D = 1$ and $D = 2$.

Lastly, note that, unlike the depth-1 case, one can no longer find a precise condition such that a $b \neq 0$ solution exists for a general A_0. The reason is that the condition for the existence of the solution is now a high-order polynomial with quite arbitrary intermediate terms. The following proposition gives a sufficient but stronger-than-necessary condition for the existence of a nontrivial solution, when all the σ_i, intermediate width d_i and regularization strength γ_i are the same.[4]

Proposition 3. *Let $\sigma_i^2 = \sigma^2 > 0$, $d_i = d_0$ and $\gamma_i = \gamma > 0$ for all i. Assuming $a_{\min} > 0$, the only solution is trivial if*

$$\frac{D+1}{2D}||\mathbb{E}[xy]||d_0^{D-1}\left(\frac{(D-1)||\mathbb{E}[xy]||}{2Dd_0(\sigma^2+d_0)^D a_{\min}}\right)^{\frac{D-1}{D+1}} < \gamma. \tag{14}$$

Nontrivial solutions exist if

$$\frac{D+1}{2D}||\mathbb{E}[xy]||d_0^{D-1}\left(\frac{(D-1)||\mathbb{E}[xy]||}{2Dd_0(\sigma^2+d_0)^D a_{\max}}\right)^{\frac{D-1}{D+1}} \geq \gamma. \tag{15}$$

Moreover, the nontrivial solutions are both lower and upper-bounded:[5]

$$\frac{1}{d_0}\left[\frac{\gamma}{||\mathbb{E}[xy]||}\right]^{\frac{1}{D-1}} \leq b^* \leq \left[\frac{||\mathbb{E}[xy]||}{d_0(\sigma^2+d_0)^D a_{\max}}\right]^{\frac{1}{D+1}}. \tag{16}$$

Proof Sketch. The proof follows from the observation that the l.h.s. of Eq. (13) is a continuous function and must cross the r.h.s. under certain sufficient conditions. □

One should compare the general condition here with the special condition for $D = 1$. One sees that for $D \geq 2$, many other factors (such as the width, the depth, and the spectrum of the data covariance A_0) come into play to determine the existence of a solution apart from the signal strength $\mathbb{E}[xy]$ and the regularization strength γ.

[4] This is equivalent to setting $c_0 = \sqrt{d_0}$. The result is qualitatively similar but involves additional factors of c_0 if σ_i, d_i, and γ_i all take different values. We thus only present the case when σ_i, d_i, and γ_i are the same for notational concision and for emphasizing the most relevant terms. Also, note that this proposition gives a *sufficient and necessary* condition if $A_0 = \sigma_x^2 I$ is proportional to the identity.

[5] For $D = 1$, we define the lower-bound as $\lim_{\eta\to 0^+}\lim_{D\to 1^+}\frac{1}{d_0}\left[\frac{\gamma+\eta}{||\mathbb{E}[xy]||}\right]^{\frac{1}{D-1}}$, which equal to zero if $\mathbb{E}[xy] \geq \gamma$, and ∞ if $\mathbb{E}[xy] < \gamma$. With this definition, this proposition applies to a two-layer net as well.

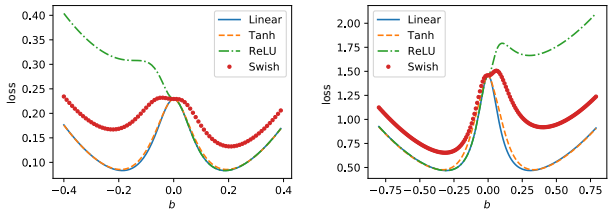

Figure 2: The training loss as a function of b for a $D = 1$ network with different activation functions in the hidden layer. For simplicity, dropout is not implemented. The non-linear activation functions we considered are ReLU, Tanh, and Swish. The left and right panels use different data. **Left:** X are Gaussian random vectors, and $y = v \cdot x$ is a linear function of x. **Right:** x are Gaussian random vectors, and $y = v \cdot tanh(x)$ are nonlinear functions of data; the weight v is obtained as a Gaussian random vector.

4.2 Which Solution is the Global Minimum?

Again, we set $\gamma_i = \gamma > 0$, $\sigma_i^2 = \sigma^2 > 0$ and $d_i = d_0 > 0$ for all i for notational concision. Using this condition and applying Lemma 3 to Theorem 2, the solution now takes the following form, where $b \geq 0$,

$$\begin{cases} U = \sqrt{d_0} b \mathbf{r}_D; \\ W^{(i)} = b \mathbf{r}_i \mathbf{r}_{i-1}^T; \\ W^{(1)} = \mathbf{r}_1 \mathbb{E}[xy]^T d_0^{D-\frac{1}{2}} b^D \left[d_0^D (\sigma^2 + d_0)^D b^{2D} A_0 + \gamma \right]^{-1}. \end{cases} \tag{17}$$

The following theorem gives a sufficient condition for the global minimum to be nontrivial. It also shows that the landscape of the linear net becomes complicated and can contain more than 1 local minimum.

Theorem 3. *Let $\sigma_i^2 = \sigma^2 > 0$, $d_i = d_0$ and $\gamma_i = \gamma > 0$ for all i and assuming $a_{\min} > 0$. Then, if*

$$||\mathbb{E}[xy]||^2 \geq \frac{\gamma^{\frac{D+1}{D}} D^2 (\sigma^2 + d_0)^{D-1} a_{max}^{\frac{D-1}{D}}}{d_0^{D-1} (D-1)^{\frac{D-1}{D}}} \tag{18}$$

the global minimum of Eq. (9) is one of the nontrivial solutions.

While there are various ways this bound can be improved, it is general enough for our purpose. In particular, one sees that, for a general depth, the condition for having a nontrivial global minimum depends not only on the $\mathbb{E}[xy]$ and γ but also on the model architecture in general. For a more general architecture with different widths etc., the architectural constant c_0 from Eq. (13) will also enter the equation. In the limit of $D \to 1^+$, relation (18) reduces to

$$||\mathbb{E}[xy]||^2 \geq \gamma^2, \tag{19}$$

which is the condition derived for the 2-layer case.

5 Implications

Relevance to nonlinear models. We first caution the readers that the following discussion should be taken with a caveat and is based on the philosophy that deep linear nets can approximate the nonlinear ones. This approximation certainly holds for fully connected models with differentiable activation functions such as tanh or Swish because they are, up to first-order Taylor expansion, a deep linear net around zero, which is the region for which our theory is the most relevant. We empirically demonstrate that close to the origin, the landscape of linear nets can indeed approximate that of nonlinear nets quite well. To compare, we plug in the solution in Theorem 4 to both linear and nonlinear models of the same architecture and compare the loss values at different values of b around $b = 0$. For simplicity, we only consider the case $D = 1$. The activation functions we consider are ReLU, Tanh, and Swish (Ramachandran et al., 2017), a modern and differentiable variant of ReLU. See Fig. 2.

The regressor $x \in \mathbb{R}^d$ is sampled as Gaussian random vectors. We consider two methods of generating y; the first one (left) is $y = v \cdot x$. The second one (right) is $y = v \cdot \tanh(x)$, where the weight $v \in \mathbb{R}^d$ is obtained as a Gaussian random vector. Fig. 2 shows that the landscape consisting of Tanh is always close to the linear landscape. Swish is not as good as Tanh, but the Swish landscape shows a similar tendency to the linear landscape. The ReLU landscape is not so close to the linear landscape

either for $b > 0$ or $b < 0$, but it agrees completely with the linear landscape on the other side, as expected. Besides the quantitative closeness, it is also important to note that all the landscapes agree qualitatively, containing the same number of local minima at similar values of b.

Landscape of multi-layer neural networks. The combination of Theorem 3 and Proposition 2 shows that the landscape of a deep neural network can become highly nontrivial when there is a weight decay and when the depth of the model is larger than 2. This gives an incomplete but meaningful picture of a network's complicated but interesting landscape beyond two layers (see Figure 1 for an incomplete summary of our results). In particular, even when the nontrivial solution is the global minimum, the trivial solution is still a local minimum that needs to be escaped. Our result suggests the previous understanding that all local minima of a deep linear net are global cannot generalize to many practical settings where deep learning is found to work well. For example, a series of works attribute the existence of bad (non-global) minima to the use of nonlinearities (Kawaguchi, 2016) or the use of a non-regular (non-differentiable) loss function (Laurent and Brecht, 2018). Our result, in contrast, shows that the use of a simple weight decay is sufficient to create a bad minimum.[6] Moreover, the problem with such a minimum is two-fold: (1) (optimization) it is not global and so needs to be "overcome" and (2) (generalization) it is a minimum that has not learned any feature at all because the model constantly outputs zero. To the best of our knowledge, previous to our work, there has not been any proof that a bad minimum can generically exist in a rather arbitrary network without any restriction on the data.[7] Thus, our result offers direct and solid theoretical justification for the widely believed importance of escaping local minima in the field of deep learning (Kleinberg et al., 2018; Liu et al., 2021; Mori et al., 2022). In particular, previous works on escaping local minima often hypothesize landscapes that are of unknown relevance to an actual neural network. With our result, this line of research can now be established with respect to landscapes that are actually deep-learning-relevant.

Previous works also argue that having a deeper depth does not create a bad minimum (Lu and Kawaguchi, 2017). While this remains true, its generality and applicability to practical settings now also seem low. Our result shows that as long as weight decay is used, and as long as $D \geq 2$, there is indeed a bad local minimum at 0. In contrast, there is no bad minimum at 0 for a depth-2 network: the point $b = 0$ is either a saddle or the global minimum.[8] Having a deeper depth thus alters the qualitative nature of the landscape, and our results agree better with the common observation that a deeper network is harder, if not impossible, to optimize.

We note that our result can also be relevant for more modern architectures such as the ResNet. Using ResNet, one needs to change the dimension of the hidden layer after every bottleneck, and a learnable linear transformation is applied here. Thus, the "effective depth" of a ResNet would be roughly between the number of its bottlenecks and its total number of blocks. For example, a ResNet18 applied to CIFAR10 often has five bottlenecks and 18 layers in total. We thus expect it to have qualitatively similar behavior to a deep linear net with a depth in between. See Figure 1. The experimental details are given in Section A.

Learnability of a neural network. Now we analyze the solution when D tends to infinity. We first note that the existence condition bound in (15) becomes exponentially harder to satisfy as D becomes large:

$$||\mathbb{E}[xy]||^2 \geq 4d_0^2 a_{\max} \gamma e^{D \log[(\sigma^2+d_0)/d_0]} + O(1). \tag{20}$$

When this bound is not satisfied, the given neural network cannot learn the data. Recall that for a two-layer net, the existence condition is nothing but $||\mathbb{E}[xy]||^2 > \gamma^2$, independent of the depth, width, or stochasticity in the model. For a deeper network, however, every factor comes into play, and the

[6] Some previous works do suggest the existence of bad minima when weight decay is present, but no direct proof exists yet. For example, Taghvaei et al. (2017) shows that when the model is approximated by a linear dynamical system, regularization can cause bad local minima. Mehta et al. (2021) shows the existence of bad local minima in deep linear networks with weight decay through numerical simulations.

[7] In the case of nonlinear networks without regularization, a few works proved the existence of bad minima. However, the previous results strongly depend on the data and are rather independent of architecture. For example, one major assumption is that the data cannot be perfected and fitted by a linear model (Yun et al., 2018; Liu, 2021; He et al., 2020). Some other works explicitly construct data distribution (Safran and Shamir, 2018; Venturi et al., 2019). Our result, in contrast, is independent of the data.

[8] Of course, in practice, the model trained with SGD can still converge to the trivial solution even if it is a saddle point (Ziyin et al., 2021) because minibatch SGD is, in general, not a good estimator of the local minima.

architecture of the model has a strong (and dominant) influence on the condition. In particular, a factor that increases polynomially in the model width and exponentially in the model depth appears.

A practical implication is that the use of weight decay may be too strong for deep networks. If one increases the depth or width of the model, one should also roughly decrease γ according to Eq. (20).

Insufficiency of the existing initialization schemes. We have shown that 0 is often a bad local minimum for deep learning. Our result further implies that escaping this local minimum can be highly practically relevant because standard initialization schemes are trapped in this local minimum for tasks where the signal $\mathbb{E}[xy]$ is weak. See Inequality (16): any nontrivial global minimum is lower-bounded by a factor proportional to $(\gamma/||\mathbb{E}[xy]||^{1/(D-1)})/d_0$, which can be seen as an approximation of the radius of the local minimum at the origin. In comparison, standard deep learning initialization schemes such as Kaiming init. initialize at a radius roughly $1/\sqrt{d_0}$. Thus, for tasks $\mathbb{E}[xy] \ll \gamma/\sqrt{d_0}$, these initialization methods are likely to initialize the model in the basin of attraction of the trivial regime, which can cause a serious failure in learning. To demonstrate, we perform a numerical simulation shown in the right panel of Figure 3, where we train $D = 2$ nonlinear networks with width 32 with SGD on tasks with varying $||\mathbb{E}[xy]||$. For sufficiently small $||\mathbb{E}[xy]||$, the model clearly is stuck at the origin.[9] In contrast, linear regression is never stuck at the origin. Our result thus suggests that it may be desirable to devise initialization methods that are functions of the data distribution.

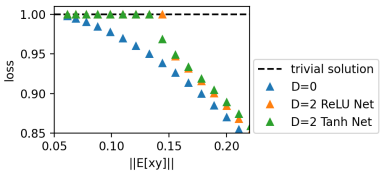

Figure 3: Training loss of $D = 2$ neural networks with ReLU and Tanh activations across synthetic tasks with different $||\mathbb{E}[xy]||$. We see that with the Kaiming initialization, both the Tanh net and the ReLU net are stuck at the trivial solution in expectation of our theory. In contrast, an optimized linear regressor ($D = 0$) is better than the trivial solution when $||\mathbb{E}[xy]|| > 0$. See Section A for experimental details.

Prediction variance of stochastic nets. A major extension of the standard neural networks is to make them stochastic, namely, to make the output a random function of the input. In a broad sense, stochastic neural networks include neural networks trained with dropout (Srivastava et al., 2014; Gal and Ghahramani, 2016), Bayesian networks (Mackay, 1992), variational autoencoders (VAE) (Kingma and Welling, 2013), and generative adversarial networks (Goodfellow et al., 2014). Stochastic networks are thus of both practical and theoretical importance to study. Our result can also be used for studying the theoretical properties of stochastic neural networks. Here, we present a simple application of our general solution to analyze the properties of a stochastic net. The following theorem summarizes our technical results.

Theorem 4. *Let $\sigma_i^2 = \sigma^2 > 0$, $d_i = d_0$ and $\gamma_i = \gamma > 0$ for all i. Let $A_0 = \sigma_x^2 I$. Then, at any global minimum of Eq. (9), holding other parameters fixed,*

1. *in the limit of large d_0, $Var[f(x)] = O\left(d_0^{-1}\right)$;*
2. *in the limit of large σ^2, $Var[f(x)] = O\left(\frac{1}{(\sigma^2)^D}\right)$;*
3. *In the limit of large D, $Var[f(x)] = O\left(e^{-2D\log[(\sigma^2+d_0)/d_0]}\right)$.*

Interestingly, the scaling of prediction variance in asymptotic σ^2 is different for different widths. The third result shows that the prediction variance decreases exponentially fast in D. In particular, this result answers a question recently proposed in Ziyin et al. (2022b): does a stochastic net trained on MSE have a prediction variance that scales towards 0? We improve on their result in the case of a deep linear net by (a) showing that the d_0^{-1} is tight in general, independent of the depth or other factors of the model, and (b) proving a bound showing that the variance also scales towards zero as depth increases, which is a novel result of our work. Our result also offers an important insight into the cause of the vanishing prediction variance. Previous works (Alemi et al., 2018) often attribute the cause to the fact that a wide neural network is too expressive. However, our result implies that this is not always the case because a linear network with limited expressivity can also have a vanishing variance as the model tends to an infinite width.

Collapses in deep learning. Lastly, we comment briefly on the apparent similarity between different types of collapses that occur in deep learning. For neural collapse, our result agrees with the recent

[9]There are many natural problems where the signal is extremely weak. One well-known example is the problem of future price prediction in finance, where the fundamental theorem of finance forbids a large $||\mathbb{E}[xy]||$ (Fama, 1970).

works that identify weight decay as a main cause (Rangamani and Banburski-Fahey, 2022). For Bayesian deep learning, Wang and Ziyin (2022) identified the cause of the posterior collapse in a two-layer VAE structure to be that the regularization of the mean of the latent variable z is too strong. More recently, the origin and its stability have also been discussed as the dimensional collapse in self-supervised learning (Ziyin et al., 2022a). Although appearing in different contexts of deep learning, the three types of collapses share the same phenomenology that the model converges to a "collapsed" regime where the learned representation becomes low-rank or constant, which agrees with the behavior of the trivial regime we identified. We refer the readers to Ziyin and Ueda (2022) for a study of how the second-order phase transition framework of statistical physics can offer a possible unified explanation of these phenomena.

6 Conclusion

In this work, we derived the exact solution of a deep linear net with arbitrary depth and width and with stochasticity. Our work sheds light on the highly complicated landscape of a deep neural network. Compared to the previous works that mostly focus on the qualitative understanding of the linear net, our result offers a more precise quantitative understanding of deep linear nets. Quantitative understanding is one major benefit of knowing the exact solution, whose usefulness we have also demonstrated with the various implications. The results, although derived for linear models, are also empirically shown to be relevant for networks with nonlinear activations. Lastly, our results strengthen the line of thought that analytical approaches to deep linear models can be used to understand deep neural networks, and it is the sincere hope of the authors to attract more attention to this promising field.

Acknowledgement

Ziyin is financially supported by the GSS scholarship of the University of Tokyo and the JSPS fellowship. Li is financially supported by CNRS. X. Meng is supported by JST CREST Grant Number JPMJCR1912, Japan.

References

Alemi, A., Poole, B., Fischer, I., Dillon, J., Saurous, R. A., and Murphy, K. (2018). Fixing a broken ELBO. In *International Conference on Machine Learning*, pages 159–168. PMLR.

Baldi, P. and Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58.

Cavazza, J., Morerio, P., Haeffele, B., Lane, C., Murino, V., and Vidal, R. (2018). Dropout as a low-rank regularizer for matrix factorization. In *International Conference on Artificial Intelligence and Statistics*, pages 435–444. PMLR.

Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B., and LeCun, Y. (2015a). The loss surfaces of multilayer networks. In *Artificial Intelligence and Statistics*, pages 192–204.

Choromanska, A., LeCun, Y., and Arous, G. B. (2015b). Open problem: The landscape of the loss surfaces of multilayer networks. In *Conference on Learning Theory*, pages 1756–1760. PMLR.

Fama, E. F. (1970). Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gotmare, A., Keskar, N. S., Xiong, C., and Socher, R. (2018). A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*.

Hardt, M. and Ma, T. (2016). Identity matters in deep learning. *arXiv preprint arXiv:1611.04231*.

Hastie, T., Montanari, A., Rosset, S., and Tibshirani, R. J. (2019). Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*.

He, F., Wang, B., and Tao, D. (2020). Piecewise linear activations substantially shape the loss surfaces of neural networks. *arXiv preprint arXiv:2003.12236*.

Kawaguchi, K. (2016). Deep learning without poor local minima. *Advances in Neural Information Processing Systems*, 29:586–594.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kleinberg, B., Li, Y., and Yuan, Y. (2018). An alternative view: When does sgd escape local minima? In *International Conference on Machine Learning*, pages 2698–2707. PMLR.

Krogh, A. and Hertz, J. A. (1992). A simple weight decay can improve generalization. In *Advances in neural information processing systems*, pages 950–957.

Laurent, T. and Brecht, J. (2018). Deep linear networks with arbitrary loss: All local minima are global. In *International conference on machine learning*, pages 2902–2907. PMLR.

Liu, B. (2021). Spurious local minima are common for deep neural networks with piecewise linear activations. *arXiv preprint arXiv:2102.13233*.

Liu, K., Ziyin, L., and Ueda, M. (2021). Noise and fluctuation of finite learning rate stochastic gradient descent.

Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Lu, H. and Kawaguchi, K. (2017). Depth creates no bad local minima. *arXiv preprint arXiv:1702.08580*.

Lucas, J., Tucker, G., Grosse, R., and Norouzi, M. (2019). Don't Blame the ELBO! A Linear VAE Perspective on Posterior Collapse.

Mackay, D. J. C. (1992). *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology.

Mehta, D., Chen, T., Tang, T., and Hauenstein, J. (2021). The loss surface of deep linear networks viewed through the algebraic geometry lens. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Mianjy, P. and Arora, R. (2019). On dropout and nuclear norm regularization. In *International Conference on Machine Learning*, pages 4575–4584. PMLR.

Mori, T., Ziyin, L., Liu, K., and Ueda, M. (2022). Power-law escape rate of SGD.

Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions.

Rangamani, A. and Banburski-Fahey, A. (2022). Neural collapse in deep homogeneous classifiers and the role of weight decay. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4243–4247. IEEE.

Safran, I. and Shamir, O. (2018). Spurious local minima are common in two-layer relu neural networks. In *International conference on machine learning*, pages 4433–4441. PMLR.

Saxe, A. M., McClelland, J. L., and Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Taghvaei, A., Kim, J. W., and Mehta, P. (2017). How regularization affects the critical points in linear networks. *Advances in neural information processing systems*, 30.

Tanaka, H., Kunin, D., Yamins, D. L., and Ganguli, S. (2020). Pruning neural networks without any data by iteratively conserving synaptic flow. *Advances in Neural Information Processing Systems*, 33:6377–6389.

Tian, Y. (2022). Deep contrastive learning is provably (almost) principal component analysis. *arXiv preprint arXiv:2201.12680*.

Venturi, L., Bandeira, A. S., and Bruna, J. (2019). Spurious valleys in one-hidden-layer neural network optimization landscapes. *Journal of Machine Learning Research*, 20:133.

Wang, Z. and Ziyin, L. (2022). Posterior collapse of a linear latent variable model. *arXiv preprint arXiv:2205.04009*.

Yun, C., Sra, S., and Jadbabaie, A. (2018). Small nonlinearities in activation functions create bad local minima in neural networks. *arXiv preprint arXiv:1802.03487*.

Ziyin, L., Li, B., Simon, J. B., and Ueda, M. (2021). SGD with a Constant Large Learning Rate Can Converge to Local Maxima.

Ziyin, L., Lubana, E. S., Ueda, M., and Tanaka, H. (2022a). What shapes the loss landscape of self-supervised learning? *arXiv preprint arXiv:2210.00638*.

Ziyin, L. and Ueda, M. (2022). Exact phase transitions in deep learning. *arXiv preprint arXiv:2205.12510*.

Ziyin, L. and Wang, Z. (2022). Sparsity by Redundancy: Solving L_1 with a Simple Reparametrization. *arXiv preprint arXiv:2210.01212*.

Ziyin, L., Zhang, H., Meng, X., Lu, Y., Xing, E., and Ueda, M. (2022b). Stochastic neural networks with infinite width are deterministic.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The experiments are only for demonstration and are straightforward to reproduce following the theory.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] The fluctuations are visually negligible.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] They are done on a single 3080Ti GPU.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Experimental Details

For the experiment in Figure 3, the input data consists of 1000 data points sampled from a multivariate Gaussian distribution: $x \sim \mathcal{N}(0, I_5)$. The target is generated by a linear transformation $y = v \cdot x$, where the norm of v is rescaled to obtain different values of $\|\mathbb{E}[xy]\|$ as the control parameter of the simulation. The models are with $D = 2$ neural networks with bias terms and with hidden width 32 for both hidden layers. The training proceeds with gradient descent with a learning rate of 0.1 for 10^4 iterations when the training loss has stopped decreasing for all the experiments.

For the CIFAR10 experiments, we train a standard ResNet18 with roughly 10^7 parameters under the standard procedure, with a batch size of 256 for 100 epochs.[10] For the linear models, we use a hidden width of 32 without any bias term. The training proceeds with SGD with batch size 256 for 100 epochs with a momentum of 0.9. The learning rate is 0.002, chosen as the best learning rate from a grid search over $[0.001, 0.002, ..., 0.01]$.

B Proofs

B.1 Proof of Lemma 1

Proof. Note that the first term in the loss function is invariant to the following rescaling for any $a > 0$:

$$\begin{cases} U_i \to aU_i; \\ W_{ij} \to W_{ij}/a; \end{cases} \tag{21}$$

meanwhile, the L_2 regularization term changes as a changes. Therefore, the global minimum must have a minimized a with respect to any U and W.

One can easily find the solution:

$$a^* = \arg\min_a \left(\gamma_u a^2 U_i^2 + \gamma_w \sum_j \frac{W_{ij}^2}{a^2} \right) = \left(\frac{\gamma_w \sum_j W_{ij}^2}{\gamma_u U_i^2} \right)^{1/4}. \tag{22}$$

Therefore, at the global minimum, we must have $\gamma_u a^2 U_i^2 = \gamma_w \sum_j \frac{W_{ij}^2}{a^2}$, so that

$$(U_i^*)^2 = (a^* U_i)^2 = \frac{\gamma_w}{\gamma_u} \sum_j (W_{ij}^*)^2, \tag{23}$$

which completes the proof. □

B.2 Proof of Lemma 2

Proof. By Lemma 1, we can write U_i as b_i and $W_{i:}$ as $b_i w_i$ where w_i is a unit vector, and finding the global minimizer of Eq. (2) is equivalent to finding the minimizer of the following objective,

$$\mathbb{E}_{x,\varepsilon} \left[\left(\sum_{i,j} b_i^2 \epsilon_i w_{ij} x_j - y \right)^2 \right] + (\gamma_u + \gamma_w) \|b\|_2^2, \tag{24}$$

$$= \mathbb{E}_x \left[\left(\sum_{i,j} b_i^2 w_{ij} x_j - y \right)^2 \right] + \sigma^2 \sum_{ij} b_i^4 \left(\sum_k w_{ik} x_k \right)^2 + (\gamma_u + \gamma_w) \|b\|_2^2, \tag{25}$$

The lemma statement is equivalent to $b_i = b_j$ for all i and j.

We prove this by contradiction. Suppose there exist i and j such that $b_i \neq b_j$, we can choose i to be the index of b_i with maximum b_i^2, and let j be the index of b_j with minimum b_j^2. Now, we can construct a different solution by the following replacement of $b_i w_{i:}$ and $b_j w_{j:}$:

$$\begin{cases} b_i^2 w_{i:} \to c^2 v; \\ b_j^2 w_{j:} \to c^2 v, \end{cases} \tag{26}$$

[10] Specifically, we use the implementation and training procedure of https://github.com/kuangliu/pytorch-cifar, with standard augmentations such as random crop, etc.

where c is a positive scalar and v is a unit vector such that $2c^2 v = b_i^2 w_{i:} + b_j^2 w_{j:}$. Note that, by the triangular inequality, $2c^2 \le b_i^2 + b_j^2$. Meanwhile, all the other terms, b_k for $k \neq i$ and $k \neq j$, are left unchanged. This transformation leaves the first term in the loss function (25) unchanged, and we now show that it decreases the other terms.

The change in the second term is

$$\left(b_i^2 \sum_k w_{ik} x_k\right)^2 + \left(b_j^2 \sum_k w_{jk} x_k\right)^2 \to 2\left(c^2 \sum_k v_k x_k\right)^2 = \frac{1}{2}\left(b_i^2 \sum_k w_{ik} x_k + b_j^2 \sum_k w_{jk} x_k\right)^2. \quad (27)$$

By the inequality $a^2 + b^2 \ge (a+b)^2/2$, we see that the left-hand side the larger than the right-hand side.

We now consider the L_2 regularization term. The change is

$$(\gamma_u + \gamma_w)(b_i^2 + b_j^2) \to 2(\gamma_u + \gamma_w)c^2, \quad (28)$$

and the left-hand side is again larger than the right-hand side by the inequality mentioned above: $2c^2 \le b_i^2 + b_j^2$. Therefore, we have constructed a solution whose loss is strictly smaller than that of the global minimum: a contradiction. Thus, the global minimum must satisfy

$$U_i^2 = U_j^2 \quad (29)$$

for all i and j.

Likewise, we can show that $U_i W_{i:} = U_j W_{j:}$ for all i and j. This is because the triangular inequality $2c^2 \le b_i^2 + b_j^2$ is only an equality if $U_i W_{i:} = U_j W_{j:}$. If $U_i W_{i:} \neq U_j W_{j:}$, following the same argument above, we arrive at another contradiction. □

B.3 Proof of Theorem 1

Proof. By Lemma 2, at any global minimum, we can write $U_* = b\mathbf{r}$ for some $b \in \mathbb{R}$. We can also write $W_* = \mathbf{r}v^T$ for a general vector $v \in \mathbb{R}^d$. Without loss of generality, we assume that $b > 0$ (because the sign of b can be absorbed into $\mathbf{r}$).

The original problem in Eq. (2) is now equivalently reduced following problem because $\mathbf{r}^T\mathbf{r} = d_1$:

$$\min_{b,v} \mathbb{E}_x\left[\left(bd_1 \sum_j v_j x_j - y\right)^2 + b^2 d_1 \sigma^2 \left(\sum_k v_k x_k\right)^2\right] + \gamma_u d_1 b^2 + \gamma_w d_1 \|v\|_2^2. \quad (30)$$

For any fixed b, the global minimum of v is well known:[11]

$$v = b\mathbb{E}[xy]^T\left[b^2\left(\sigma^2 + d_1\right)A_0 + \gamma_w I\right]^{-1}. \quad (31)$$

By Lemma 1, at a global minimum, b also satisfies the following condition:

$$b^2 = \frac{\gamma_w}{\gamma_u}\|v\|^2, \quad (32)$$

One solution to this equation is $b = 0$, and we are interested in whether solutions with $b \neq 0$ exist. If there is no other solution, then $b = 0$ must be the unique global minimum; otherwise, we need to identify which of the solutions are actual global minima. When $b \neq 0$,

$$\left\|\left[b^2\left(\sigma^2 + d_1\right)A_0 + \gamma_w I\right]^{-1}\mathbb{E}[xy]\right\|^2 = \frac{\gamma_u}{\gamma_w}. \quad (33)$$

Note that the left-hand side is monotonically decreasing in b^2, and is equal to $\gamma_w^{-2}\|\mathbb{E}[xy]\|^2$ when $b = 0$. When $b \to \infty$, the left-hand side tends to 0. Because the left-hand side is a continuous and monotonic function of b, a unique solution $b_* > 0$ that satisfies Eq. (33) exists if and only if $\gamma_w^{-2}\|\mathbb{E}[xy]\|^2 > \gamma_u/\gamma_w$, or,

$$\|\mathbb{E}[xy]\|^2 > \gamma_u\gamma_w. \quad (34)$$

[11] Namely, it is the solution of a ridgeless linear regression problem.

Therefore, at most, three candidates for global minima of the loss function exist:

$$\begin{cases} b = 0,\ v = 0 & \text{if } \|\mathbb{E}[xy]\|^2 \le \gamma_u\gamma_w; \\ b = \pm b_*,\ v = b\left[b^2\left(\sigma^2 + d_1\right)A_0 + \gamma_w I\right]^{-1}\mathbb{E}[xy], & \text{if } \|\mathbb{E}[xy]\|^2 > \gamma_u\gamma_w, \end{cases} \tag{35}$$

where $b^* > 0$.

In the second case, one needs to discern the saddle points from the global minima. Using the expression of v, one finds the expression of the loss function as a function of b

$$d_1(d_1+\sigma^2)b^4\sum_i \frac{\mathbb{E}[x'y]_i^2 a_i}{[b^2(\sigma^2+d_1)a_i+\gamma_w]^2} - 2b^2 d_1 \sum_i \frac{\mathbb{E}[x'y]_i^2}{b^2(\sigma^2+d_1)a_i+\gamma_w} + \mathbb{E}[y^2] + \gamma_u d_1 b^2 + \gamma_w d_1 \sum_i \frac{\mathbb{E}[x'y]_i^2 b^2}{[b^2(\sigma^2+d_1)a_i+\gamma_w]^2}, \tag{36}$$

where $x' = Rx$ such that RA_0R^{-1} is a diagonal matrix. We now show that condition (34) is sufficient to guarantee that 0 is not the global minimum.

At $b = 0$, the first nonvanishing derivative of b is the second-order derivative. The second order derivative at $b = 0$ is

$$-2d_1\|\mathbb{E}[xy]\|^2/\gamma_w + 2\gamma_u d_1, \tag{37}$$

which is negative if and only if $\|\mathbb{E}[xy]\|^2 > \gamma_u\gamma_w$. If the second derivative at $b = 0$ is negative, $b = 0$ cannot be a minimum. It then follows that for $\|\mathbb{E}[xy]\|^2 > \gamma_u\gamma_w$, $b = \pm b^*$, $v = b\left[b^2\left(\sigma^2 + d_1\right)A_0 + \gamma_w I\right]^{-1}\mathbb{E}[xy]$, if $\|\mathbb{E}[xy]\|^2 > \gamma_u\gamma_w$ are the two global minimum (because the loss is invariant to the sign flip of b). For the same reason, when $\|\mathbb{E}[xy]\|^2 < \gamma_u\gamma_w$, $b = 0$ gives the unique global minimum. This finishes the proof. □

B.4 Proof of Proposition 1

Proof. We first show that there exists a constant r such that the global minimum must be confined within a (closed) r-Ball around the origin. The objective (9) can be upper-bounded by

$$Eq.\ (9) \ge \gamma_u\|U\|^2 + \sum_{i=1}^{D}\gamma_i\|W^{(i)}\|^2 \ge \gamma_{\min}\left(\|U\|^2 + \sum_i \|W^{(i)}\|^2\right), \tag{38}$$

where $\gamma_{\min} := \min_{i\in\{u,1,2,\ldots,D\}} > 0$. Now, let w denote be the union of all the parameters $(U, W^{(i)})$ and viewed as a vector. We see that the above inequality is equivalent to

$$Eq.\ (9) \ge \gamma_{\min}\|w\|^2. \tag{39}$$

Now, note that the loss value at the origin is $\mathbb{E}[y^2]$, which means that for any w, whose norm $\|w\|^2 \ge \mathbb{E}[y^2]/\gamma_{\min}$, the loss value must be larger than the loss value of the origin. Therefore, let $r = \mathbb{E}[y^2]/\gamma_{\min}$, we have proved that the global minimum must lie in a closed r-Ball around the origin.

As the last step, because the objective is a continuous function of w and the r-Ball is a compact set, the minimum of the objective in this r-Ball is achievable. This completes the proof. □

B.5 Proof of Theorem 2

We divide the proof into the proof of a proposition and a lemma, and combining the following proposition and lemma obtains the theorem statement.

B.5.1 Proposition 4

Proposition 4. *Any global minimum of Eq. (9) is of the form*

$$\begin{cases} U = b_u\mathbf{r}_D; \\ W^{(i)} = b_i\mathbf{r}_i\mathbf{r}_{i-1}^T; \\ W^{(1)} = \mathbf{r}_1\mathbb{E}[xy]^T(b_u\prod_{i=2}^D b_i)\mu\left[(b_u\prod_{i=2}^D b_i)^2 s^2\left(\sigma^2+d_1\right)A_0 + \gamma_w I\right]^{-1}, \end{cases} \tag{40}$$

where $\mu = \prod_{i=2}^D d_i$, $s^2 = \prod_{i=2}^D d_i(\sigma^2 + d_i)$, $b_u \ge 0$ *and* $b_i \ge 0$, *and* $\mathbf{r}_i = (\pm 1, \ldots, \pm 1)$ *is an arbitrary vertex of a* d_i*-dimensional hypercube for all* i.

Proof. Note that the trivial solution is also a special case of this solution with $b = 0$. We thus focus on deriving the form of the nontrivial solution.

We prove by induction on D. The base case with depth 1 is proved in Theorem 1. We now assume that the same holds for depth $D-1$ and prove that it also holds for depth D.

For any fixed $W^{(1)}$, the loss function can be equivalently written as

$$\mathbb{E}_{\tilde{x}}\mathbb{E}_{\epsilon^{(2)},\ldots,\epsilon^{(D)}}\left(\sum_{i_1,i_2,\ldots,i_D}^{d_1,d_2,\ldots d_D} U_{i_D}\epsilon^{(D)}_{i_D}\ldots\epsilon^{(2)}_{i_2}W^{(2)}_{i_2 i_1}\tilde{x}_{i_1} - y\right)^2 + \gamma_u\|U\|^2 + \sum_{i=2}^{D}\gamma_i\|W^{(i)}\|^2 + const., \tag{41}$$

where $\tilde{x} = \epsilon^{(1)}_{i_1}\sum_i W^{(1)}_{i_1 i}x_i$. Namely, we have reduced the problem to a problem involving only a depth $D-1$ linear net with a transformed input $\tilde{x}$.

By the induction assumption, the global minimum of this problem takes the form of Eq. (10), which means that the loss function can be written in the following form:

$$\mathbb{E}_{\tilde{x}}\mathbb{E}_{\epsilon^{(2)},\ldots,\epsilon^{(D)}}\left(b_u b_D\ldots b_3\sum_{i_1,i_2,\ldots,i_D}^{d_1,d_2,\ldots d_D}\epsilon^{(D)}_{i_D}\ldots\epsilon^{(2)}_{i_2}v_{i_1}\tilde{x}_{i_1} - y\right)^2 + L_2\ reg., \tag{42}$$

for an arbitrary optimizable vector v_{i_1}. The term $\sum_{i_2,\ldots,i_D}^{d_2,\ldots d_D}\epsilon^{(D)}_{i_D}\ldots\epsilon^{(2)}_{i_2} := \eta$ can now be regarded as a single random variable such that $\mathbb{E}[\eta] = \prod_{i=2}^{D} d_i := \mu$ and $\mathbb{E}[\eta^2] = \prod_{i=2}^{D} d_i(\sigma_i^2 + d_i) := s^2$. Computing the expectation over all the noises except for $\epsilon^{(1)}$, one finds

$$\mathbb{E}_{\tilde{x}}\left(b_u b_D\ldots b_3 s\sum_{i_1} v_{i_1}\tilde{x}_{i_1} - \frac{\mu y}{s}\right)^2 + L_2\ reg. + const. \tag{43}$$

$$= \mathbb{E}_{x,\epsilon^{(1)}}\left(b_u b_D\ldots b_3 s\sum_{i,i_1} v_{i_1}\epsilon^{(1)}_{i_1}W^{(1)}_{i_1 i}x_i - \frac{\mu y}{s}\right)^2 + L_2\ reg. + const., \tag{44}$$

where we have ignored the constant term because it does not affect the minimizer of the loss. Namely, we have reduced the original problem to a two-layer linear net problem where the label becomes effectively rescaled for a deep network.

For any fixed $b_u, \ldots, b_3$, we can define $\bar{x} := b_u b_D\ldots b_3 s x$, and obtain the following problem, whose global minimum we have already derived:

$$\mathbb{E}_{\bar{x}}\mathbb{E}_{\epsilon^2,\ldots,\epsilon_D}\left(\sum_{i,i_1} v_{i_1}W^{(1)}_{i_1 i}\bar{x}_i - \frac{\mu y}{s}\right)^2. \tag{45}$$

By Theorem 1, the global minimum is identically 0 if $\|\mathbb{E}[\mu\bar{x}y/s]\|^2 < d_2\gamma_2\gamma_1$, or, $\mathbb{E}[xy] \le \frac{\gamma_2\gamma_1}{b_3^2\ldots b_u^2(\prod_{i=3}^{D} d_i)}$. When $\mathbb{E}[xy] > \frac{\gamma_2\gamma_1}{b_3^2\ldots b_u^2(\prod_{i=3}^{D} d_i)}$, the solution can be non-trivial:

$$\begin{cases} v_* = b_2^*\mathbf{r}_1; \\ W_* = \mathbf{r}_1\mathbb{E}[xy]^T\mu b_2^* b_3\ldots b_u\left[(b_2^*)^2 d_3^2\ldots d_D^2 b_u^2 s^2\left(\sigma^2 + d_1\right)A_0 + \gamma_1 I\right]^{-1}, \end{cases} \tag{46}$$

for some b_2^*. This proves the theorem. □

B.6 Lemma 3

Lemma 3. *At any global minimum of Eq. (9), let* $b_1 := \sqrt{\|W_{i:}\|^2/d}$ *and* $b_{D+1} := b_u$,

$$\gamma_{k+1}d_{k+1}b_{k+1}^2 = \gamma_k d_{k-1}b_k^2. \tag{47}$$

Proof. It is sufficient to show that for all k and i,

$$\gamma_{k+1}\sum_{ij}(W^{k+1}_{ji})^2 = \gamma_k\sum_{ij}(W^k_{ij})^2. \tag{48}$$

We prove by contradiction. Let U^*, W^* be the global minimum of the loss function. Assuming that for an arbitrary k,

$$\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2 \neq \gamma_k \sum_{ij}(W_{ij}^{*,k})^2. \tag{49}$$

Introduce W^a such that $W_{ji}^{a,k+1} = aW_{ji}^{*,k+1}$ and $W_{ji}^{a,k} = W_{ji}^{*,k}/a$. The loss without regularization is invariant under the transformation of $W^* \to W^a$, namely

$$L_0(W^*) = L_0(W^a). \tag{50}$$

In the regularization, all the terms remain invariant except two terms:

$$\begin{cases}\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2 \to \gamma_{k+1}\sum_{ij}(W_{ji}^{a,k+1})^2 = a^2\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2 \\ \gamma_k \sum_{ij}(W_{ij}^{*,k})^2 \to \gamma_k \sum_{ij}(W_{ji}^{a,k})^2 = a^{-2}\gamma_k \sum_{ij}(W_{ji}^{*,k})^2\end{cases} \tag{51}$$

It could be shown that, the sum of $a^2\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2$ and $a^{-2}\gamma_k\sum_{ij}(W_{ji}^{*,k})^2$ reaches its minimum when $a^2 = \sqrt{\frac{\gamma_k\sum_{ij}(W_{ji}^{*,k})^2}{\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2}}$. If $\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2 \neq \gamma_k\sum_{ij}(W_{ij}^{*,k})^2$, one can choose a to minimize the regularization terms in the loss function such that $L(W^a) < L(W^*)$, indicating W^* is not the global minimum. Thus, $\gamma_{k+1}\sum_{ij}(W_{ji}^{*,k+1})^2 \neq \gamma_k\sum_{ij}(W_{ij}^{*,k})^2$ cannot be true. □

B.7 Proof of Proposition 2

Proof. Let

$$L_0 = \mathbb{E}_{\tilde{x}}\mathbb{E}_{\epsilon^2,\ldots,\epsilon_D}\left(\sum_{i_1,i_2,\ldots,i_D}^{d_1,d_2,\ldots d_D} U_{i_D}\epsilon_{i_D}^{(D)}\ldots\epsilon_{i_1}^{(1)}W_{i_1 i}^{(1)}x_i - y\right)^2. \tag{52}$$

L_0 is a polynomial containing $2D$ + 2th order, D + 1th order, and 0th order terms in terms of parameters U and W. The second order derivative of L is thus a polynomial containing $2D$-th order and $(D-1)$-th order terms; however, other orders are not possible. For $D \geq 2$, there are no constant terms in the Hessian of L, and there is at least a parameter in each of the terms.

The Hessian of the full loss function with regularization is

$$\frac{\partial^2 L}{\partial^2 U_i U_j} = \frac{\partial^2 L_0}{\partial^2 U_i U_j} + (1-\delta_{ij})2\gamma_u(U_i + U_j) + \delta_{ij}2\gamma_u; \tag{53}$$

$$\frac{\partial^2 L}{\partial^2 W_{jk}^i U_l} = \frac{\partial^2 L_0}{\partial^2 W_{jk}^i U_l} + 2(\gamma_w W_{jk}^i + \gamma_u U_l); \tag{54}$$

$$\frac{\partial^2 L}{\partial^2 W_{jk}^i W_{mn}^l} = \frac{\partial^2 L_0}{\partial^2 W_{jk}^i W_{mn}^l} + (1-\delta_{il}\delta_{jm}\delta_{kn})2\gamma_w(W_{jk}^i + W_{mn}^l) + \delta_{il}\delta_{jm}\delta_{kn}2\gamma_w. \tag{55}$$

For $U = 0$, $W = 0$, the Hessian of L_0 is 0, since each term in L_0 contains at least a U or a W. The Hessian of L becomes

$$\left.\frac{\partial^2 L}{\partial^2 U_i U_j}\right|_{U,W=0} = \delta_{ij}2\gamma_u; \tag{56}$$

$$\left.\frac{\partial^2 L}{\partial^2 W_{jk}^i U_l}\right|_{U,W=0} = 0; \tag{57}$$

$$\left.\frac{\partial^2 L}{\partial^2 W_{jk}^i W_{mn}^l}\right|_{U,W=0} = \delta_{il}\delta_{jm}\delta_{kn}2\gamma_w. \tag{58}$$

The Hessian of L is a positive-definite matrix. Thus, $U = 0$, $W = 0$ is always a local minimum of the loss function L. □

B.8 Proof of Proposition 3

We first apply Lemma 3 to determine the condition for the nontrivial solution to exist. In particular, the Lemma must hold for $W^{(2)}$ and $W^{(1)}$, which leads to the following condition:

$$\|b^{D-1}d_0^{D-1}[b^{2D}d_0^D(\sigma^2+d_0)^D A_0+\gamma]^{-1}\mathbb{E}[xy]\|^2 = 1. \tag{59}$$

Note that the left-hand side is a continuous function that tends to 0 as $b \to \infty$. Therefore, it is sufficient to find the condition that guarantees that there exists b such that the l.h.s. is larger than 1. For any b, the l.h.s. is a monotonically decreasing function of any eigenvalue of A_0, and so the following two inequalities hold:

$$\begin{cases} \|b^{D-1}d_0^{D-1}(b^{2D}d_0^D(\sigma^2+d_0)^D\sigma_x^2+\gamma)^{-1}\mathbb{E}[xy]\| \le \|b^{D-1}d_0^{D-1}(b^{2D}d_0^D(\sigma^2+d_0)^D a_{\min}+\gamma)^{-1}\mathbb{E}[xy]\| \\ \|b^{D-1}d_0^{D-1}(b^{2D}d_0^D(\sigma^2+d_0)^D\sigma_x^2+\gamma)^{-1}\mathbb{E}[xy]\| \ge \|b^{D-1}d_0^{D-1}(b^{2D}d_0^D(\sigma^2+d_0)^D a_{\max}+\gamma)^{-1}\mathbb{E}[xy]\|. \end{cases} \tag{60}$$

The second inequality implies that if

$$\|b^{D-1}d_0^{D-1}[b^{2D}d_0^D(\sigma^2+d_0)^D a_{\max}+\gamma]^{-1}\mathbb{E}[xy]\| > 1, \tag{61}$$

a nontrivial solution must exist. This condition is equivalent to the existence of a b such that

$$d_0^D(\sigma^2+d_0)^D a_{\max} b^{2D} - \|\mathbb{E}[xy]\|b^{D-1}d_0^{D-1} < -\gamma, \tag{62}$$

which is a polynomial inequality that does not admit an explicit condition for b for a general D. Since the l.h.s is a continuous function that increases to infinity as $b \to \infty$, one sufficient condition for (62) to hold is that the minimizer of the l.h.s. is smaller than γ.

Note that the left-hand side of Eq. (62) diverges to ∞ as $b \to \pm\infty$ and tends to zero as $b \to 0$. Moreover, Eq. (62) is lower-bounded and must have a nontrivial minimizer for some $b > 0$ because the coefficient of the b^{D-1} term is strictly negative. One can thus find its minimizer by taking derivative. In particular, the left-hand side is minimized when

$$b^{D+1} = \frac{(D-1)\|\mathbb{E}[xy]\|}{2Dd_0(\sigma^2+d_0)^D a_{\max}}, \tag{63}$$

and we can obtain the following sufficient condition for (62) to be satisfiable, which, in turn, implies that (59) is satisfiable:

$$\frac{D+1}{2D}\|\mathbb{E}[xy]\|d_0^{D-1}\left(\frac{(D-1)\|\mathbb{E}[xy]\|}{2Dd_0(\sigma^2+d_0)^D a_{\max}}\right)^{\frac{D-1}{D+1}} > \gamma, \tag{64}$$

which is identical to the proposition statement in (15).

Now, we come back to condition (60) to derive a sufficient condition for the trivial solution to be the only solution. The first inequality in Condition (60) implies that if

$$\|b^{D-1}d_0^{D-1}[b^{2D}d_0^D(\sigma^2+d_0)^D a_{\min}+\gamma]^{-1}\mathbb{E}[xy]\| \le 1, \tag{65}$$

the only possible solution is the trivial one, and the condition for this to hold can be found using the same procedure as above to be

$$\frac{D+1}{2D}\|\mathbb{E}[xy]\|d_0^{D-1}\left(\frac{(D-1)\|\mathbb{E}[xy]\|}{2Dd_0(\sigma^2+d_0)^D a_{\min}}\right)^{\frac{D-1}{D+1}} \le \gamma, \tag{66}$$

which is identical to (14).

We now prove the upper bound for the solution in ((16)). Because for any b, the first condition in 60 gives an upper bound, and so any b that makes the upper bound less than 1 cannot be a solution. This means that any b for which

$$\|b^{D-1}d_0^{D-1}[b^{2D}d_0^D(\sigma^2+d_0)^D a_{\min}+\gamma]^{-1}\mathbb{E}[xy]\| \le 1 \tag{67}$$

cannot be a solution. This condition holds if and only if

$$d_0^D(\sigma^2+d_0)^D a_{\min} b^{2D} - \|\mathbb{E}[xy]\|b^{D-1}d_0^{D-1} > -\gamma. \tag{68}$$

Because $\gamma > 0$, one sufficient condition to ensure this is that there exists b such that

$$d_0(\sigma^2 + d_0)^D a_{\min} b^{2D} - ||\mathbb{E}[xy]|| b^{D-1} > 0, \tag{69}$$

which is equivalent to

$$b > \left[\frac{||\mathbb{E}[xy]||}{d_0(\sigma^2 + d_0)^D a_{\min}} \right]^{\frac{1}{D+1}}. \tag{70}$$

Namely, any solution b^* satisfies

$$b^* \leq \left[\frac{||\mathbb{E}[xy]||}{d_0(\sigma^2 + d_0)^D a_{\min}} \right]^{\frac{1}{D+1}}. \tag{71}$$

We can also find a lower bound for all possible solutions. When $D > 1$, another sufficient condition for Eq. (68) to hold is that there exists b such that

$$||\mathbb{E}[xy]|| d_0^{D-1} b^{D-1} < \gamma. \tag{72}$$

because the b^{2D} term is always positive. This condition then implies that any solution must satisfy:

$$b^* \geq \frac{1}{d_0} \left[\frac{\gamma}{||\mathbb{E}[xy]||} \right]^{\frac{1}{D-1}}. \tag{73}$$

For $D = 1$, we have by Theorem 1 that

$$b^* > 0 \tag{74}$$

if and only if $\mathbb{E}[xy] > \gamma$. This means that

$$b^* \geq \lim_{\eta \to 0^+} \lim_{D \to 1^+} \frac{1}{d_0} \left[\frac{\gamma + \eta}{||\mathbb{E}[xy]||} \right]^{\frac{1}{D-1}} = \begin{cases} \infty & \text{if } \mathbb{E}[xy] \geq \gamma; \\ 0 & \text{if } \mathbb{E}[xy] < \gamma. \end{cases} \tag{75}$$

This finishes the proof. □

B.9 Proof of Theorem 3

Proof. When nontrivial solutions exist, we are interested in identifying when $b = 0$ is not the global minimum. To achieve this, we compare the loss of $b = 0$ with the other solutions. Plug the trivial solution into the loss function in Eq. (9), the loss is easily identified to be $L_{\text{trivial}} = E[y^2]$.

For the nontrivial minimum, defining f to be the model,

$$f(x) := \sum_{i, i_1, i_2, \ldots, i_D}^{d, d_1, d_2, \ldots d_D} U_{i_D} \epsilon_{i_D}^{(D)} \ldots \epsilon_{i_2}^{(2)} W_{i_2 i_1}^{(2)} \epsilon_{i_1}^{(1)} W_{i_1 i}^{(1)} x \tag{76}$$

$$= \eta d_0^D b^{2D} \mathbb{E}[xy]^T [b^{2D} d_0^D (\sigma^2 + d_0)^D A_0 + \gamma I]^{-1} x, \tag{77}$$

where, similar to the previous proof, we have defined $\sum_{i_1, \ldots, i_D}^{d_1, \ldots d_D} \epsilon_{i_D}^{(D)} \ldots \epsilon_{i_1}^{(1)} := \eta$ such that $\mathbb{E}[\eta] = \prod_i^D d_i = d_0^D$ and $\mathbb{E}[\eta^2] = \prod_i^D d_i(\sigma_i^2 + d_i) := d_0^D(\sigma^2 + d_0)^D$. With this notation, The loss function becomes

$$\mathbb{E}_x \mathbb{E}_\eta (f(x) - y)^2 + L_2 \; reg. \tag{78}$$

$$= \mathbb{E}_{x,\eta}[f(x)^2] - 2\mathbb{E}_{x,\eta}[y f(x)] + \mathbb{E}_x[y^2] + L_2 \; reg. \tag{79}$$

$$= \sum_i \frac{d_0^{3D} (\sigma^2 + d_0)^D b^{4D} a_i \mathbb{E}[x'y]_i^2}{[d_0^D (\sigma^2 + d_0)^D a_i b^{2D} + \gamma]^2} - 2 \sum_i \frac{d_0^{2D} b^{2D} \mathbb{E}[x'y]_i^2}{d_0^D (\sigma^2 + d_0)^D a_i b^{2D} + \gamma} + \mathbb{E}_x[y^2] + L_2 \; reg. \tag{80}$$

The last equation is obtained by rotating x using a orthogonal matrix such that $R^{-1} A_0 R = \text{diag}(a_i)$ and denoting the rotated x as $x' = Rx$. With x', The L_2 reg term takes the form of

$$L_2 \; reg. = \gamma D d_0^2 b^2 + \gamma \sum_i \frac{d_0^{2D} b^{2D} ||\mathbb{E}[x'y]_i||^2}{(d_0^D (\sigma^2 + d_0)^D b^{2D} a_i + \gamma)^2}. \tag{81}$$

Combining the expressions of (81) and (80), we obtain that the difference between the loss at the non-trivial solution and the loss at 0 is

$$-\sum_i \frac{d_0^{2D} b^{2D} \mathbb{E}[x'y]_i^2}{[d_0^D(\sigma^2+d_0)^D a_i b^{2D}+\gamma]} + \gamma D d_0^2 b^2. \tag{82}$$

Satisfaction of the following relation thus guarantees that the global minimum is nontrivial:

$$\sum_i \frac{d_0^{2D} b^{2D} \mathbb{E}[x'y]_i^2}{[d_0^D(\sigma^2+d_0)^D a_i b^{2D}+\gamma]} \geq \gamma D d_0^2 b^2. \tag{83}$$

This relation is satisfied if

$$\frac{d_0^{2D} b^{2D} ||\mathbb{E}[xy]||^2}{[d_0^D(\sigma^2+d_0)^D a_{max} b^{2D}+\gamma]} \geq \gamma D d_0^2 b^2 \tag{84}$$

$$\frac{b^{2D-2}}{[d_0^D(\sigma^2+d_0)^D a_{max} b^{2D}+\gamma]} \geq \frac{\gamma D}{d_0^{2D-2}||\mathbb{E}[xy]||^2}. \tag{85}$$

$$\tag{86}$$

The derivative or l.h.s. with respect to b is

$$\frac{b^{2D-3}[(2D-2)\gamma - 2d_0^D(\sigma^2+d_0)^D a_{max} d^{2D}]}{[d_0^D(\sigma^2+d_0)^D a_{max} b^{2D}+\gamma]^2}. \tag{87}$$

For $b, \gamma \in (0,\infty)$, the derivative dives below 0, indicating the l.h.s. of (86) has a global maximum at a strictly positive b. The value of b is found when setting the derivative to 0, namely

$$\frac{b^{2D-3}[(2D-2)\gamma - 2d_0^D(\sigma^2+d_0)^D a_{max} d^{2D}]}{[d_0^D(\sigma^2+d_0)^D a_{max} b^{2D}+\gamma]^2} = 0 \tag{88}$$

$$(2D-2)\gamma - 2d_0^D(\sigma^2+d_0)^D a_{max} d^{2D} = 0 \tag{89}$$

$$b^{2D} = \frac{(D-1)\gamma}{d_0^D(\sigma^2+d_0)^D a_{max}}. \tag{90}$$

The maximum value then takes the form

$$\frac{(D-1)^{\frac{D-1}{D}}}{D\gamma^{\frac{1}{D}} d_0^{D-1}(\sigma^2+d_0)^{D-1} a_{max}^{\frac{D-1}{D}}}. \tag{91}$$

The following condition thus guarantees that the global minimum is non-trivial

$$\frac{(D-1)^{\frac{D-1}{D}}}{D\gamma^{\frac{1}{D}} d_0^{D-1}(\sigma^2+d_0)^{D-1} a_{max}^{\frac{D-1}{D}}} \geq \frac{\gamma D}{d_0^{2D-2}||\mathbb{E}[xy]||^2} \tag{92}$$

$$||\mathbb{E}[xy]||^2 \geq \frac{\gamma^{\frac{D+1}{D}} D^2 (\sigma^2+d_0)^{D-1} a_{max}^{\frac{D-1}{D}}}{d_0^{D-1}(D-1)^{\frac{D-1}{D}}}. \tag{93}$$

This finishes the proof. □

B.10 Proof of Theorem 4

Proof. The model prediction is:

$$f(x) := \sum_{i,i_1,i_2,\ldots,i_D}^{d,d_1,d_2,\ldots d_D} U_{i_D} \epsilon_{i_D}^{(D)} \ldots \epsilon_{i_2}^{(2)} W_{i_2 i_1}^{(2)} \epsilon_{i_1}^{(1)} W_{i_1 i}^{(1)} x \tag{94}$$

$$= \eta d_0^D b^{2D} \mathbb{E}[xy]^T [b^{2D} d_0^D (\sigma^2+d_0)^D \sigma_x^2 I + \gamma I]^{-1} x. \tag{95}$$

One can find the expectation value and variance of a model prediction:

$$\mathbb{E}_\eta[f(x)] = \frac{d_0^{2D} b^{2D} \mathbb{E}[xy]^T x}{b^{2D} d_0^D (\sigma^2+d_0)^D \sigma_x^2 + \gamma} \tag{96}$$

For the trivial solution, the theorem is trivially true. We thus focus on the case when the global minimum is nontrivial.

The variance of the model is

$$Var[f(x)] = \mathbb{E}[f(x)^2] - \mathbb{E}[f(x)]^2 \tag{97}$$

$$= \frac{(\sigma^2 + d_0)^D d_0^{3D} b^{4D} (\mathbb{E}[xy]^T x)^2}{[b^{2D} d_0^D (\sigma^2 + d_0)^D \sigma_x^2 + \gamma]^2} - \frac{d_0^{4D} b^{4D} (\mathbb{E}[xy]^T x)^2}{[b^{2D} d_0^D (\sigma^2 + d_0)^D]^2 \sigma_x^2 + \gamma]^2} \tag{98}$$

$$= \frac{d_0^{3D}[(\sigma^2 + d_0)^D - d_0^D] b^{4D} (\mathbb{E}[xy]^T x)^2}{[b^{2D} d_0^D (\sigma^2 + d_0)^D \sigma_x^2 + \gamma]^2} \tag{99}$$

$$= \frac{d_0^{3D}[(\sigma^2 + d_0)^D - d_0^D] b^{2D+2} (\mathbb{E}[xy]^T x)^2}{\|\mathbb{E}[xy]\|^2}, \tag{100}$$

where the last equation follows from Eq. (13). The variance can be upper-bounded by applying (16),

$$Var[f(x)] \leq \frac{d_0^D[(\sigma^2 + d_0)^D - d_0^D](\mathbb{E}[xy]^T x)^2}{(\sigma^2 + d_0)^{2D} \sigma_x^2} \propto \frac{d_0^D[(\sigma^2 + d_0)^D - d_0^D]}{(\sigma^2 + d_0)^{2D}}. \tag{101}$$

We first consider the limit $d_0 \to \infty$ with fixed σ^2:

$$Var[f(x)] \propto \frac{D d_0^{2D-1} \sigma^2}{(d_0 + \sigma^2)^{2D}} = O\left(\frac{1}{d_0}\right). \tag{102}$$

For the limit $\sigma^2 \to \infty$ with d_0 fixed, we have

$$Var[f(x)] = O\left(\frac{1}{(\sigma^2)^D}\right). \tag{103}$$

Additionally, we can consider the limit when $D \to \infty$ as we fix both σ^2 and d_0:

$$Var[f(x)] = O\left(e^{-D 2 \log[(\sigma^2 + d_0)/d_0]}\right), \tag{104}$$

which is an exponential decay. □

C Exact Form of b^* for $D = 1$

Note that our main result does not specify the exact value of b^*. This is because b^* must satisfy the condition in Eq. (6), which is equivalent to a high-order polynomial in b with coefficients being general functions of the eigenvalues of A_0, whose solutions are generally not analytical by Galois theory. One special case where an analytical formula exists for b is when $A_0 = \sigma_x^2 I$. Practically, this can be achieved for any (full-rank) dataset if we disentangle and rescale the data by the whitening transformation: $x \to \sigma_x \sqrt{A_0^{-1}} x$. In this case, we have

$$b_*^2 = \frac{\sqrt{\frac{\gamma_w}{\gamma_u}} \|\mathbb{E}[xy]\| - \gamma_w}{(\sigma^2 + d_1)\sigma_x^2}, \tag{105}$$

and

$$v = \pm \sqrt{\frac{\sqrt{\frac{\gamma_u}{\gamma_w}} \|\mathbb{E}[xy]\| - \gamma_u}{\sigma_x^2 (\sigma^2 + d_1)}} \frac{\mathbb{E}[xy]}{\|\mathbb{E}[xy]\|}, \tag{106}$$

where $v = W_{i:}$.

D Effect of Bias

This section studies a deep linear network with biases for every layer and compares it with the no-bias networks. We first study a general case when the data does not receive any preprocessing. We then show that the problem reduces to the setting we considered in the main text under the common data preprocessing schemes that centers the input and output data: $\mathbb{E}[x] = 0$, and $\mathbb{E}[y] = 0$.

D.1 Two-layer network

The two-layer linear network with bias is defined as

$$f_b(x; U, W, \beta^U, \beta^W) = \sum_i \epsilon_i U_i (W_{i:} \cdot x + \beta_i^W) + \beta^U, \tag{107}$$

where $\beta^W \in \mathbb{R}^{d_1}$ is the bias in the hidden layer, and $\beta^U \in \mathbb{R}$ is the bias at the output layer. The loss function is

$$L_b(U, W, \beta^U, \beta^W) = \mathbb{E}_{\epsilon,x,y}[(\sum_i \epsilon_i U_i (W_{i:} \cdot x + \beta_i^W) + \beta^U - y)^2] + L_2 \tag{108}$$

$$= \mathbb{E}_{x,y}\left[(UWx + U\beta^W + \beta^U - y)^2 + \sigma^2 \sum_i U_i^2 (W_{i:} \cdot x + \beta_i^W)^2\right] \tag{109}$$

$$+ \gamma_u(\|U\|^2 + (\beta^U)^2) + \gamma_w(\|W\|^2 + \|\beta^W\|^2). \tag{110}$$

It is helpful to concatenate x and 1 into a single vector $x' := (x, 1)^{\mathrm{T}}$ and concatenate W and β^W into a single matrix W' such that W, β^W, x, and W', x' are related via the following equation

$$Wx + \beta^W = W'x'. \tag{111}$$

Using W' and x', the model can be written as

$$f_b(x', U, W', \beta^U) = \sum_i \epsilon_i U_i W'_{i:} \cdot x' + \beta^U. \tag{112}$$

The loss function simplifies to

$$L_b(U, W', \beta) = \mathbb{E}_{\epsilon,x,y}[(\sum_i \epsilon_i U_i W'_{i:} \cdot x' + \beta^U - y)^2] + \gamma_u(\|U\|^2 + (\beta^U)^2) + \gamma_w\|W'\|^2. \tag{113}$$

Note that (113) contains similar rescaling invariance between U and W' and the invariance of aligning $W'_{i:}$ and $W'_{j:}$. One can thus obtain the following two propositions that mirror Lemma 1 and 2.

Proposition 5. *At the global minimum of* (108), $U_j^2 = \frac{\gamma_w}{\gamma_u}\left(\sum_i W_{ji}^2 + (\beta_j^W)^2\right)$.

Proposition 6. *At the global minimum, for all i and j, we have*

$$\begin{cases} U_i^2 = U_j^2; \\ U_i W_{i:} = U_j W_{j:}; \\ U_i \beta_i^W = U_j \beta_j^W. \end{cases} \tag{114}$$

The proofs are omitted because they are the same as those of Lemma 1 and 2, substituting W by W'. By following a procedure similar to finding the solution for a no-bias network, one finds that

Theorem 5. *The global minimum of Eq.* (108) *is of the form*

$$\begin{cases} U = b\mathbf{r}; \\ \beta^U = \frac{d_1 b\left[(d_1+\sigma^2)\frac{\gamma_u}{\gamma_w} b - 1\right] v\mathbb{E}[x] - \left(d_1 \frac{\gamma_u}{\gamma_w} b^2 - 1\right)\mathbb{E}[y]}{(d_1+\sigma^2) d_1 \frac{\gamma_u^2}{\gamma_w^2} b^4 + (\gamma_u - 2) d_1 \frac{\gamma_u}{\gamma_w} b^2 + \gamma_u + 1}; \\ W = \mathbf{r}b\left\{\mathbb{E}[x]\left[b\frac{\gamma_u}{\gamma_w}(d_1 + b\sigma^2) - 1\right]\beta^U + \mathbb{E}[xy]\right\}^T [b^2(d_1+\sigma^2)A_0 + \gamma_w I]^{-1}; \\ \beta^W = -\mathbf{r}\frac{\gamma_u}{\gamma_w} b\beta^U, \end{cases} \tag{115}$$

where b satisfies

$$\gamma_u b^2 = b^2 \frac{\gamma_w (\frac{\mathbb{E}[y]S_1S_3}{S_4}\mathbb{E}[x] - \mathbb{E}[xy])(M^{-1})^2 (\frac{\mathbb{E}[y]S_1S_3}{S_4}\mathbb{E}[x] - \mathbb{E}[xy])^T + \frac{\gamma_u^2}{\gamma_w}\left(\frac{S_3}{S_4}\mathbb{E}[y] - b\frac{S_2}{S_4}\mathbb{E}[x]M^{-1}\mathbb{E}[xy]\right)^2}{\left(b\frac{S_2S_1}{S_4}\mathbb{E}[x]M^{-1}\mathbb{E}[x]^T - 1\right)^2}, \tag{116}$$

where M, S_1, S_2, S_3, S_4 are functions of the model parameters and b, defined in Eq. (122).

Proof. First of all, we derive a handy relation satisfied by β^U and β^W at all the stationary points. The zero-gradient condition of the stationary points gives

$$\begin{cases} \mathbb{E}_{x,y}[2(UWx + U\beta^W + \beta^U - y)]U + 2\gamma_w\beta^W = 0; \\ \mathbb{E}_{x,y}[2(UWx + U\beta^W + \beta^U - y)] + 2\gamma_u\beta^U = 0, \end{cases} \tag{117}$$

leading to

$$U\gamma_u\beta^U + \gamma_w\beta^W = 0 \tag{118}$$

$$\beta_i^W = -\frac{\gamma_u}{\gamma_w}U_i\beta^U. \tag{119}$$

Proposition 5 and proposition 6 implies that we can define $b := |U_i|$ and $bv := U_iW_{i:}$. Consequently, $U_i\beta_i^W = -\frac{\gamma_u}{\gamma_w}b^2\beta^U$, and the loss function can be written as

$$\mathbb{E}_x\left[\left(bd_1\sum_j v_jx_j - (d_1\frac{\gamma_u}{\gamma_w}b^2 - 1)\beta^U - y\right)^2 + b^2d_1\sigma^2\left(\sum_k v_kx_k - \frac{\gamma_u}{\gamma_w}b\beta^U\right)^2\right] + \gamma_ud_1b^2 + \gamma_wd_1\|v\|_2^2 + \gamma_u\left(\frac{b^2d_1\gamma_u}{\gamma_w} + 1\right)(\beta^U)^2. \tag{120}$$

The respective zero-gradient condition for v and β^U implies that for all stationary points,

$$\begin{cases} v = \left[b^2(d_1 + \sigma^2)A_0 + \gamma_wI\right]^{-1}b\left\{\mathbb{E}[x]\left[b\frac{\gamma_u}{\gamma_w}(d_1 + b\sigma^2) - 1\right]\beta^U + \mathbb{E}[xy]\right\}; \\ \beta^U = \frac{d_1b\left[(d_1+\sigma^2)\frac{\gamma_u}{\gamma_w}b-1\right]v\mathbb{E}[x]-\left(d_1\frac{\gamma_u}{\gamma_w}b^2-1\right)\mathbb{E}[y]}{(d_1+\sigma^2)d_1\frac{\gamma_u^2}{\gamma_w^2}b^4+(\gamma_u-2)d_1\frac{\gamma_u}{\gamma_w}b^2+\gamma_u+1}. \end{cases} \tag{121}$$

To shorten the expressions, we introduce

$$\begin{cases} M = b^2(d_1 + \sigma^2)A_0 + \gamma_wI; \\ S_1 = b\frac{\gamma_u}{\gamma_w}(d_1 + b\sigma^2) - 1; \\ S_2 = d_1b\left[(d_1 + \sigma^2)\frac{\gamma_u}{\gamma_w}b - 1\right]; \\ S_3 = d_1\frac{\gamma_u}{\gamma_w}b^2 - 1; \\ S_4 = (d_1 + \sigma^2)d_1\frac{\gamma_u^2}{\gamma_w^2}b^4 + (\gamma_u - 2)d_1\frac{\gamma_u}{\gamma_w}b^2 + \gamma_u + 1. \end{cases} \tag{122}$$

With M, S_1, S_2, S_3, S_4, we have

$$\begin{cases} v = M^{-1}b(\mathbb{E}[x]S_1\beta^U + \mathbb{E}[xy]); \\ \beta^U = \frac{S_2v\mathbb{E}[x]-S_3\mathbb{E}[y]}{S_4}. \end{cases} \tag{123}$$

The inner product of v and $\mathbb{E}[x]$ can be solved as

$$v\mathbb{E}[x] = b\frac{\frac{S_3}{S_4}\mathbb{E}[x]M^{-1}\mathbb{E}[x]S_1\mathbb{E}[y] - \mathbb{E}[x]M^{-1}\mathbb{E}[xy]}{b\frac{S_2}{S_4}\mathbb{E}[x]M^{-1}\mathbb{E}[x]S_1 - 1}. \tag{124}$$

Inserting the expression of $v\mathbb{E}[x]$ into the expression of β^U one obtains

$$\beta^U = \frac{S_3\mathbb{E}[y] - bS_2\mathbb{E}[x]M^{-1}\mathbb{E}[xy]}{b\mathbb{E}[x]M^{-1}\mathbb{E}[x]S_1S_2 - S_4} \tag{125}$$

The global minimum must thus satisfy

$$\gamma_ub^2 = \gamma_w\|v\|^2 + \gamma_u\frac{b^2d_1\gamma_u}{\gamma_w}(\beta^U)^2 \tag{126}$$

$$= b^2\frac{\gamma_w(\frac{\mathbb{E}[y]S_1S_3}{S_4}\mathbb{E}[x] - \mathbb{E}[xy])(M^{-1})^2(\frac{\mathbb{E}[y]S_1S_3}{S_4}\mathbb{E}[x] - \mathbb{E}[xy])^T + \frac{\gamma_u^2}{\gamma_w}\left(\frac{S_3}{S_4}\mathbb{E}[y] - b\frac{S_2}{S_4}\mathbb{E}[x]M^{-1}\mathbb{E}[xy]\right)^2}{\left(b\frac{S_2S_1}{S_4}\mathbb{E}[x]M^{-1}\mathbb{E}[x]^T - 1\right)^2}. \tag{127}$$

This completes the proof. □

Remark. *As in the no-bias case, we have reduced the original problem to a one-dimensional problem. However, the condition for b becomes so complicated that it is almost impossible to understand. That being said, the numerical simulations we have done all carry the bias terms, suggesting that even with the bias term, the mechanisms are qualitatively similar, and so the approach in the main text is justified.*

When $\mathbb{E}[x] = 0$, the solution can be simplified a little:

$$\begin{cases} U = \mathbf{r}b; \\ \beta^U = -\frac{d_1 \frac{\gamma_u}{\gamma_w} b^2 - 1}{(d_1+\sigma^2) d_1 \frac{\gamma_u^2}{\gamma_w^2} b^4 + (\gamma_u - 2) d_1 \frac{\gamma_u}{\gamma_w} b^2 + \gamma_u + 1} \mathbb{E}[y]; \\ W = \mathbf{r}b\mathbb{E}[xy]^T [b^2(d_1+\sigma^2)A_0 + \gamma_w I]^{-1}; \\ \beta^W = \mathbf{r}\frac{\gamma_u}{\gamma_w} b \frac{d_1 \frac{\gamma_u}{\gamma_w} b^2 - 1}{(d_1+\sigma^2) d_1 \frac{\gamma_u^2}{\gamma_w^2} b^4 + (\gamma_u - 2) d_1 \frac{\gamma_u}{\gamma_w} b^2 + \gamma_u + 1} \mathbb{E}[y], \end{cases} \tag{128}$$

where the value of b is either 0 or determined by

$$\gamma_u = \gamma_w |\mathbb{E}[xy]^T [b^2(d_1+\sigma^2)A_0 + \gamma_w I]^{-1}|^2 + \frac{\gamma_u^2}{\gamma_w} \mathbb{E}[y]^2 \left(\frac{d_1 \frac{\gamma_u}{\gamma_w} b^2 - 1}{(d_1+\sigma^2) d_1 \frac{\gamma_u^2}{\gamma_w^2} b^4 + (\gamma_u - 2) d_1 \frac{\gamma_u}{\gamma_w} b^2 + \gamma_u + 1} \right)^2. \tag{129}$$

In this case, the expression of W is identical to the no-bias model. The bias of both layers is proportional to $\mathbb{E}[y]$. The equation determining the value of b is also similar to the no-bias case. The only difference is the term proportional to $\mathbb{E}[y]^2$.

We also note that the solution becomes significantly simplified when $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$. This could be seen by finding the partial derivative of L with respect to β^W and β^U and then setting them to 0. When $\mathbb{E}[x] = 0$, $\mathbb{E}[y] = 0$, one obtains:

$$\begin{cases} \frac{\partial L}{\partial \beta_i^W} = U_i U \beta^W + U_i \beta^U + \gamma_w \beta_i^W = 0; \\ \frac{\partial L}{\partial \beta^U} = U\beta^W + \beta^U + \gamma_u \beta^U = 0. \end{cases} \tag{130}$$

These equations lead to

$$U\gamma_u \beta^U + \gamma_w \beta^W = 0, \tag{131}$$

implying

$$\begin{cases} \beta^U = 0; \\ \beta^W = 0. \end{cases} \tag{132}$$

In practice, it is common and usually recommended practice to subtract the average of x and y from the data and achieve precisely $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$. We generalize this result to deeper networks in the next section.

D.2 Deep linear network

Let β be a $\left(\sum_i^D d_i + 1\right)$-dimensional vector concatenating all $\beta^1, \beta^2, ..., \beta^D, \beta^U$, and denoting the collection of all the weights U, W^D, ..., W^1 by w, the model of a deep linear network with bias is defined as

$$f_b(x, W^D, ..., W^1, U, \beta^D, ..., \beta^1, \beta^U) \tag{133}$$

$$= (\epsilon^U \circ U)((\epsilon^D \circ W^D)(...((\epsilon^2 \circ W^2)((W^1 x + \beta^1) + \beta^2)...) + \beta^D) + \beta^U \tag{134}$$

$$= (\epsilon^U \circ U)(\epsilon^D \circ W^D)...(\epsilon^2 \circ W^2)W^1 x + (\epsilon^U \circ U)(\epsilon^D \circ W^D)...(\epsilon^2 \circ W^2)\beta^1 \tag{135}$$

$$+ (\epsilon^U \circ U)(\epsilon^D \circ W^D)...(\epsilon^3 \circ W^3)\beta^2 + ... + (\epsilon^U \circ U)\beta^D + \beta^U \tag{136}$$

$$= (\epsilon^U \circ U)(\epsilon^D \circ W^D)...(\epsilon^2 \circ W^2)W^1 x + bias(w, \beta), \tag{137}$$

where

$$bias(w, \beta) = (\epsilon^U \circ U)(\epsilon^D \circ W^D)...(\epsilon^2 \circ W^2)\beta^1 + (\epsilon^U \circ U)(\epsilon^D \circ W^D)...(\epsilon^3 \circ W^3)\beta^2 + ... + (\epsilon^U \circ U)\beta^D + \beta^U, \tag{138}$$

and $\circ$ denotes Hadamard product. The loss function is

$$L_b(x, y, w, \beta) = \mathbb{E}_{\epsilon,x,y}[(f_b(x, w, \beta) - y)^2] + L_2(w, \beta). \tag{139}$$

Proposition 5 and Proposition 6 can be generated to deep linear network. Similar to the no-bias case, we can reduce the landscape to a 1-dimensional problem by performing induction on D and using the 2-dimensional case as the base step. However, we do not solve this case explicitly here because the involved expressions now become too long and complicated even to write down, nor can they directly offer too much insight. We thus only focus on the case when the data has been properly preprocessed. Namely, $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$.

For simplicity, we assume that the regularization strength for all the layers employs the value γ. The following theorem shows that When $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$, the biases vanish for an arbitrarily deep linear network:

Theorem 6. *Let* $\mathbb{E}[x] = 0$ *and* $\mathbb{E}[y] = 0$. *The global minima of Eq.* (139) *have* $\beta^1 = 0, \beta^2 = 0, ..., \beta^D = 0, \beta^U = 0$.

Proof. At the global minimum, the gradient of the loss function vanishes. In particular, the derivatives with respect to β vanish:

$$\frac{\partial L_b(x, y, w, \beta)}{\partial \beta_i} = 0; \tag{140}$$

$$\mathbb{E}_{\epsilon,x,y}\left[\frac{\partial f_b(x, w, \beta)}{\partial \beta_i}(f_b(x, w, \beta) - y)\right] + \gamma\beta_i = 0; \tag{141}$$

$$\mathbb{E}_{\epsilon,x,y}\left[\frac{\partial bias(w, \beta)}{\partial \beta_i}(f_b(x, w, \beta) - y)\right] + \gamma\beta_i = 0; \tag{142}$$

$$\mathbb{E}_{\epsilon}\left[\frac{\partial bias(w, \beta)}{\partial \beta_i}(f_b(\mathbb{E}[x], w, \beta) - \mathbb{E}[y])\right] + \gamma\beta_i = 0, \tag{143}$$

where β_i is the ith element of β. The last equation is obtained since $f_b(x, w, \beta)$ is a linear function of x. Using the condition $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$, Equation (143) becomes

$$\mathbb{E}_{\epsilon}\left[\frac{\partial bias(w, \beta)}{\partial \beta_i} bias(w, \beta)\right] + \gamma\beta_i = 0. \tag{144}$$

$bias(w, \beta)$ is a linear combination of β_i. Consequently, $\partial bias(w, \beta)/\partial\beta_i$ does not depend on β, and $bias(w, \beta)\partial bias(w, \beta)/\partial\beta_i$ is a linear combination of β_i. The $\sum_i^D d_i + 1$ equation derived from vanishing gradient yield a set of $\sum_i^D d_i + 1$ linear equations of the form $M(w)\beta = 0$, where $M(w)$ is a $\left(\sum_i^D d_i + 1\right) \times \left(\sum_i^D d_i + 1\right)$ matrix with dependence on w. These linear equations are linearly independent, since the term $\partial L_2(w, \beta)/\partial\beta_i = 2\gamma\beta_i$ and is different in each of the equations. Thus, the linear system $M(w)\beta = 0$ has $\left(\sum_i^D d_i + 1\right)$ independent equations and $\left(\sum_i^D d_i + 1\right)$ variables. The only possible solution to this linear system is

$$\beta = 0. \tag{145}$$

This finishes the proof. □

Thus, for a deep linear network, a model without bias is good enough to describe data satisfying $\mathbb{E}[x] = 0$ and $\mathbb{E}[y] = 0$, which could be achieved by subtracting the mean of the data.

RÉSUMÉ

Cette thèse étudie deux cas particuliers de calcul numérique. L'un est la simulation numérique du modèle des disques durs à l'aide des Méthodes de Monte-Carlo par chaînes de Markov, et l'autre est l'apprentissage profond.
Les chapitres deux à sept concernent le modèle des disques durs. Nous parallélisons l'algorithme rapide de (straight) event-chain Monte Carlo en instaurant un critère simple qui enforce la causalité. Nous atteignons des performances supérieures grâce à la programmation lock-free et avons formellement vérifié la mise en œuvre. Nous comparons les variantes des algorithmes de event-chain Monte Carlo par des configurations des disques durs dérivées d'empilements clairsemés. Nous montrons que ces variantes, ne variant que dans le détail, s'échelonnent différemment par rapport au paramètre de relaxation de la configuration, confirmant notre théorie d'échelle. Nous nous dérivons des formules efficaces estimant la pression à partir de ses deux définitions. Avec ces formules, une mise en œuvre de pointe des algorithmes d'échantillonnage et une analyse statistique appropriée, nous évaluons la pression et sa barre d'erreur avec une précision sans précédent.
L'algorithme d'optimisation et la fonction objectif présentés dans l'apprentissage profond sont abordés dans cette thèse. Nous découvrons que l'algorithme du gradient stochastique montre des comportements indésirables tel que converger vers un maximum local ou ne pas échapper à un point col dans les modèles simplifiés que nous avons conçus. Nous trouvons l'expression exactement des minima dans un réseau linéaire profond avec décrochage et dégradation des pondérations et démontrons que, selon la profondeur, il peut y avoir un minimum indésirable à l'origine de l'espace des paramètres.

MOTS CLÉS

Physique statistique, Simulation numérique, Chaînes de Markov, Modèle des disques durs, Transition de phase, Apprentissage profond

ABSTRACT

This thesis studies two specific cases of computation. One is the simulation of the hard-disk model using Markov-chain Monte Carlo algorithms, and the other is deep learning.
Chapters two to seven concern the hard-disk model. We parallelize the fast (straight) event-chain Monte Carlo algorithm by introducing a rigorous criteria that enforces causality. We achieve superior performance through lock-free programming, and formally verify the implementation. We benchmark the variants of the event-chain Monte Carlo algorithms by hard-disk configurations derived from sparse packings. We show that these variants, varying only in detail, scale differently with respect to the relaxation parameter of the configuration, confirming our scaling theory. We derive efficient formulas estimating pressure from both of its definitions. With these formulas, state-of-the-art implementation of the sampling algorithms, and proper statistical analysis, we evaluate pressure and its error bar to unprecedented precision.
Both optimization algorithm and loss landscape featured in deep learning are discussed in this thesis. We find out that the stochastic gradient descent algorithm shows undesired behavior such as converging to a local maximum or failing to escape a saddle point in the toy models we have designed. We find the closed-form expression of minima in a deep linear network with dropout and weight decay and demonstrate that, depending on the depth, there can be an undesired minimum at the origin of parameter space.

KEYWORDS

Statistical physics, Simulation, Markov chains, Hard-disk model, Phase transition, Deep learning