Equilibration of Orientational Order in Hard Disks via Arcuate Event-Chain Monte Carlo

Master's Thesis in Physics

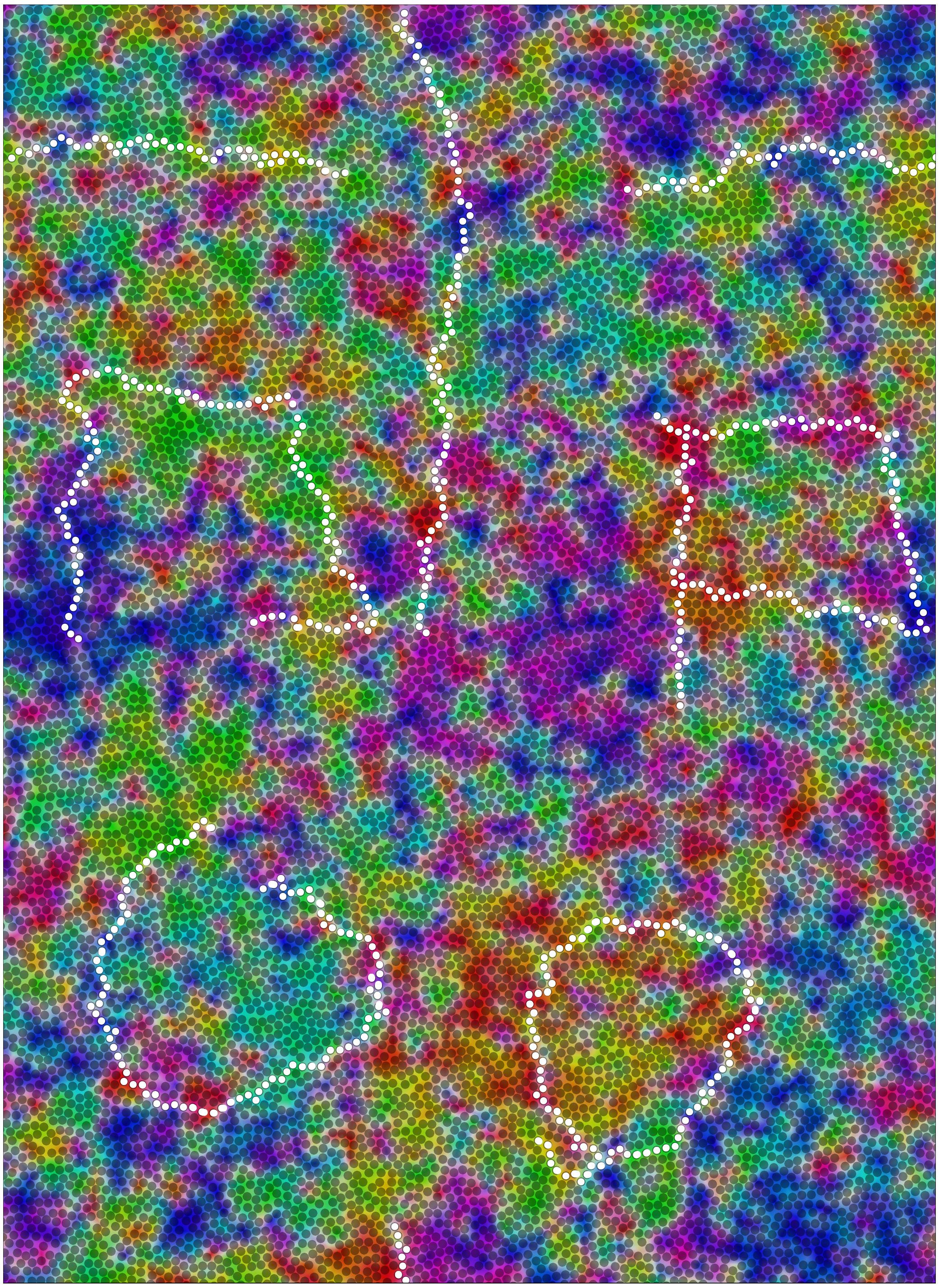
Presented by

Robert F. B. Weigel

2018–04–03

Institut für Theoretische Physik
Friedrich-Alexander-Universität Erlangen-Nürnberg

Supervisor:
Prof. Dr. Sebastian C. Kapfer


ECMC
SquareMC
ArcMC

1. Abstract

For hard disks in two dimensions the conventional Metropolis Monte Carlo algorithm exhibits dramatic slowing down in the vicinity of the liquid-hexatic and the hexatic-solid phase transition. An improved Monte Carlo algorithm, the event-chain Monte Carlo algorithm (ECMC) has facilitated extensive study of the hexatic-solid transition, by rapidly equilibrating positional order. ECMC is a lifted Markov-chain algorithm. It employs infinitesimal continuous-time moves, in which single particles are displaced with a constant velocity. An avalanche of coordinated single-particle displacements forms an event chain. This renders ECMC well-suited for equilibration of positional order. At the liquid-hexatic transition however equilibration of orientational order is the limiting factor, in which ECMC is still not satisfactory. In the related soft-disk model even a critical slowing down is observed, due to diverging orientational correlation length.

In this work we substantially generalize ECMC, in order to couple to orientational degrees of freedom. A variable velocity is obtained in two different ways: First by the introduction of velocity changes at collision events. To this end a previously unknown, generalized formulation of the global balance condition is exploited. A valid but not viable algorithm is constructed. Second, by the introduction of turning events we construct the arcuate ECMC algorithm (ArcMC). We demonstrate the correctness of ArcMC and its capability of rotating a domain of orientational correlation. Furthermore we generalize the computation of pressure in the framework of ECMC to variable velocity and derive the distribution of collision angles to be a purely geometric property. Our analysis of correlation and mixing time with respect to orientational order reveals that ECMC, ArcMC, and the Metropolis algorithm exhibit the same scaling with the number of particles, only differing by constant factors. We find that ArcMC does not exceed ECMC in equilibrating orientational order. Finally we determine the chain displacement for ECMC to be optimal, when it is greater or equal half the free length in the system.

In this project simulation code was written in C++ 11[1] and executed on computers of Institut für Theoretische Physik (chair Prof. Mecke)[2] and Regionales Rechen Zentrum Erlangen (RRZE)[3]. Data analysis was performed using Python 2.7.13[4] and gnuplot 5.0[5]. Diagonalization and Nelder-Mead optimization was performed with Wolfram Mathematica 11.2[6].
Graphics were created with gnuplot 5.0, Ti*k*Z 3.0.1a[7], and order.py[8].
Typeset by LaTeX2e[9]

[1]`http://isocpp.org`
[2]`http://www.theorie1.physik.fau.de`
[3]`http://www.rrze.fau.de/forschung/hpc`
[4]`http://www.python.org`
[5]`http://www.gnuplot.info`
[6]`http://www.wolfram.com`
[7]`http://sourceforge.net/projects/pgf`
[8]`https://github.com/Cell-veto/postlhc`, reference [1]
[9]pdfTeX 3.14159265-2.6-1.40.17 (TeX Live 2016/Debian)

Contents

1. Abstract **5**

2. Introduction **9**

3. Markov chains and lifting **13**

4. Monte Carlo algorithms for hard disks **16**
- 4.1. Metropolis Monte Carlo (MMC) 16
- 4.2. Event-chain Monte Carlo (ECMC) 17
  - 4.2.1. Calculation of pressure 19
  - 4.2.2. Distribution of the collision angle and the mean number of collisions per chain 20
  - 4.2.3. Compact initialization as a degenerate case 21

5. Design considerations of arcuate ECMC for hard disks **24**
- 5.1. Arcuate ECMC turning at collisions, with variable magnitude of velocity . 24
  - 5.1.1. Impossibility of imposing an angle 25
  - 5.1.2. ECMC turning at collisions (turn-at-collision-ECMC) 26
- 5.2. Arcuate ECMC turning between collisions 30
  - 5.2.1. Square ECMC (SquareMC) 30
  - 5.2.2. Arcuate event-chain Monte Carlo (ArcMC) 31
- 5.3. Validation 34

6. Correlation and mixing time of local Markov chains in hard disks **39**
- 6.1. Correlation time 39
- 6.2. Mixing time 42
- 6.3. Optimal chain displacement for ECMC 48

7. Conclusion **51**

References **54**

A. Appendices **55**
- A.1. Fifty-six disks in a box 55
- A.2. Cell subdivision 55
- A.3. Computations for ECMC and its variants 60
  - A.3.1. Cell-crossing event 61
  - A.3.2. Collision event 62

Acknowledgments **63**

2. Introduction

The melting transitions in $D = 2$ dimensions differ greatly from the melting of solid to liquid phase in $D = 3$. In tow-dimensional systems melting is driven by the unbinding of topological defects. For example in the two-dimensional XY model [2, 3]. The KTHNY two-step scenario [4, 5, 6] predicts the existence of a novel phase, the hexatic phase, in between the liquid and solid phase for two-dimensional systems. The two phase transitions are predicted to be of continuous type, but the liquid-hexatic transition being a first-order transition is also compatible with the scenario.

Discovered already in 1962 [7], a phase transition also exists in the hard-disk model. Since then the nature of this transition was a subject of debate. While analytical methods played a minor role, the developement of numerical simulation methods, such as the Metropolis Monte Carlo algorithm (MMC, [8]), was driven by the study of the hard-disk model. The model consists of $N \in \mathbb{N}$ circular particles with diameter σ and position $\vec{\mathbf{r}}_i$ located in a simulation box under periodic boundary conditions (PBC). The simulation box is a rectangle $\mathbb{B} = [0, L_{\mathbb{B},1}) \times [0, L_{\mathbb{B},2})$ in two-dimensional Euclidean space. The mean of the side lengths be denoted by $L_{\mathbb{B}}$. Viewing the particles and their periodic images as a crystal, the box prescribes an *orthorhombic* system. Hard disks interact via the pair potential

$$U(\vec{\mathbf{r}}_i, \vec{\mathbf{r}}_j) = U(\|\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j\|) = \begin{cases} 0, & \text{if } \|\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j\| > \sigma, \\ +\infty, & \text{otherwise,} \end{cases} \tag{1}$$

which is short-ranged and purely repulsive. In fact the pair potential is only an implementation of the non-overlap condition which applies to hard particles. Thus it is surprising that hard disks do form a solid [7] even in the absence of attractive forces. The formation of the solid is instead driven by entropic depletion force [9]. Since the pair potential of hard particles sets no energy scale, the characteristics of the system are insensitive to the inverse temperature β. The system is rather governed by the global packing fraction $\phi = N\pi\sigma^2/(4V_{\mathbb{B}})$ – the ratio of the area covered by particles to the area populated with particles. At high packing fractions above $\phi \approx 0.72$ the system is in the solid phase [10]. Since a crystal cannot be stable in $D = 2$ dimensions [11], the positional and orientational correlations are only quasi-long-ranged, which means that the corresponding correlation functions decay algebraically. Algebraical decay implies that correlations are scale free.

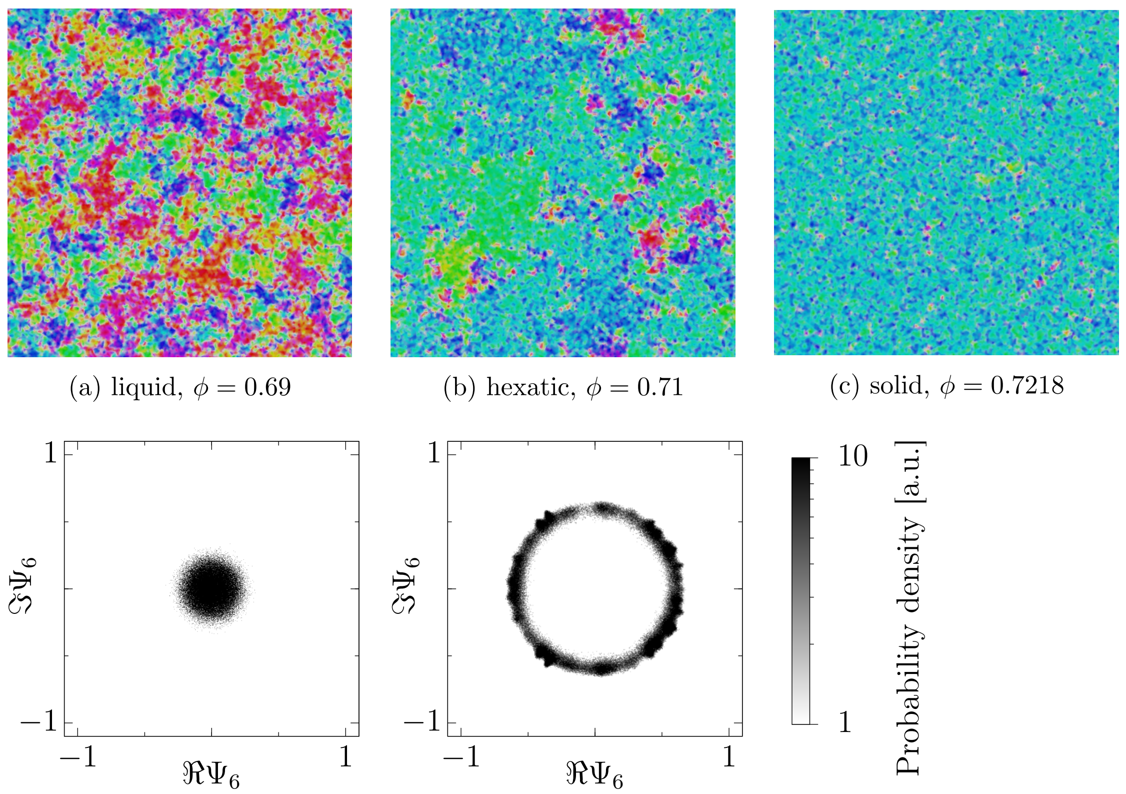
Orientational order does not address internal rotational degrees of freedom of the particles themselves, as disks are rotation-invariant. Instead orientational order is defined with respect to the particle-particle orientation. It is characterized by the global orientational order parameter Ψ_6, which is complex-valued and encodes intensity and direction of orientational order. Following [1, 12] the intensive parameter Ψ_6 is defined as

$$\Psi_6 := \frac{1}{N} \sum_i \sum_j \frac{A_{ij}}{A_i} \exp(6\mathrm{i}\theta_{ij}) = \frac{1}{N} \sum_i \psi_6(i), \tag{2}$$

where θ_{ij} is the angle between the vector $\vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i$ (respecting minimum image convention under PBC) and an arbitrary reference direction (here $\vec{\mathbf{e}}_1$). The weight A_{ij}/A_i is defined via the Voronoi tessellation constructed from the particle coordinates. A_{ij} is the length of the edge, belonging to particles i and j; with $A_{ij} = 0$, if the respective particles have no common Voronoi edge. $A_i = \sum_j A_{ij}$ is the perimeter of the Voronoi cell of particle i. The weight A_{ij}/A_i ensures that Ψ_6 is continuous with respect to variations in the particle coordinates [1]. This definition of Ψ_6 is equivalent to viewing Ψ_6 as average of the local orientational order parameter $\psi_6(i)$ of particle i, averaged over all particles. The local order parameter field $\psi_6(\vec{\mathbf{r}})$ is defined by interpolation between particle coordinates. In several figures the phase $\arg \psi_6(\vec{\mathbf{r}})$ is encoded in hue, while the magnitude $|\psi_6(\vec{\mathbf{r}})|$ is encoded in saturation, *e. g.* on page 3 or in Fig. 1c which shows a configuration in the solid phase. The maximum magnitude, $|\Psi_6| = 1$, corresponds to the perfectly ordered triangular lattice. This represents the close-packing limit of the solid phase. In the liquid phase, below $\phi \approx 0.70$ [10], positional and orientational correlation functions decay exponentially. Since exponential decay sets a length scale, the correlations are short-ranged. Fig. 1a shows the typical correlation domains in the liquid phase and in Fig. 1d the corresponding distribution of Ψ_6 is shown to be a circular cloud around $\Psi_6 = 0$. At slightly higher packing fractions $\phi \approx 0.71$ the occurrence of low intensity Ψ_6 values decreases, leading to a ring-shaped distribution, see Fig. 1e. This qualitative change is caused by system-spanning orientational correlation domains, as in Fig. 1b. This is the hexatic phase, exhibiting short-range positional and quasi-long-range orientational correlations. The nature of the phase transitions and the existence of the hexatic phase has long been debated; the alternative scenario being a single first-order liquid-solid transition. Only recently the liquid-hexatic transition was proven to be a first-order transition, while the hexatic-solid transition is of continuous type [13, 10]. Due to the first order transition, hexatic and liquid phase can coexist. The length scale of orientational correlation domains is finite in the liquid phase, while it diverges in the hexatic phase.

At a continuous phase transition the correlation length generally diverges. Upon this divergence Markov-chain Monte Carlo methods experience a critical slowing down. For example this is the case for the ferromagnetic Ising model and for the soft-disk model with a r^{-6} potential [12]. The hard-disk model does in principle not have a critical slowing down; yet the correlation length of the liquid phase is large in the vicinity of the first-order transition. In spin models the problem of critical slowing down is addressed using cluster algorithms, such as the Wolff algorithm [14]. Those employ non-local moves. For continuous models however, only few cluster algorithms are known. For example there is a chain algorithm operating with finite displacements [15] and the pivot-cluster algorithm [16]. Yet these are insufficient for studying hard-disk melting, because their efficiency drops at high packing fractions.

In contrast the event-chain Monte Carlo algorithm (ECMC, [17, 18]) consists of coherent local moves. Its integrating framework beyond MMC is the concept of lifted Markov chains [19, 20]. In ECMC a particle is displaced with a velocity $\vec{\mathbf{v}}$ in a certain direction up to the point when it collides with another particle. Then the first particle stops and the hit particle moves on with exactly the velocity $\vec{\mathbf{v}}$. When a certain amount of displacement has been accumulated, the event chain ends. The next event chain starts with a different velocity and a random first particle. On page 3, in the top row, trajectories of



(a) liquid, $\phi = 0.69$ (b) hexatic, $\phi = 0.71$ (c) solid, $\phi = 0.7218$

(d) liquid, $\phi = 0.69$ (e) hexatic, $\phi = 0.71$

short-ranged p. order	short-ranged p. order	quasi-long-ranged p. order
short-ranged o. order	quasi-long-ranged o. order	quasi-long-ranged o. order

Figure 1: (a), (b), (c) Typical order parameter field $\psi_6(\vec{\mathbf{r}})$ for $N = 32256$. In the liquid phase many small correlation domains are present, whereas in the hexatic phase one system-spanning domain coexists with liquid domains. (d), (e) Distributions of Ψ_6 for $N = 2016$. Correspondingly, highly-probable values of $|\Psi_6|$ are close to zero in the liquid phase and close to a finite value in the hexatic phase. The third row gives the characteristic range of positional and orientational order for the respective phases.

two independent event chains are shown in white (in directions $\vec{\mathbf{e}}_1$ and $\vec{\mathbf{e}}_2$, both crossing the periodic boundary). With its chain moves, ECMC is well-suited for equilibration of positional degrees of freedom. Hence this algorithm helped clarifying the nature of the hard-disk hexatic phase [10].

Still ECMC does not satisfactorily equilibrate orientational order. Therefore it stands to reason to extend ECMC with rotational 'stirring' moves, in order to couple to the orientational degrees of freedom. The construction of such a new algorithm is the aim of this project. This requires the velocity $\vec{\mathbf{v}}$ to be variable during event chains – a problem on which virtually no literature exists.

In section 3 the example of a random walk is used to review the basic concepts of Markov-chain Monte Carlo, particularly with regard to lifted Markov chains. In section 4 the description of the hard-disk formulations of MMC and ECMC is followed by the presentation of new results: We generalize the computation of pressure in the framework of ECMC to the case of varying velocity, which has not been considered previously. Moreover we derive the distribution of collision angles, which permits us to calculate the exact mean number of particle collisions per event chain. Finally a degenerate initial condition is discussed, under which ECMC may fail to converge. This degenerate case has not been explicitly reported on in previous publications. In section 5 we derive the global balance condition for a modified ECMC algorithm with the velocity changing upon particle collisions and illustrate its implications. By introducing a new type of event, we construct the algorithm SquareMC and the more general ArcMC. Their correctness is shown, and it is demonstrated that ArcMC is capable of persistently rotating a correlation domain. In section 6 the performance of MMC, ECMC, SquareMC and ArcMC is analyzed in terms of the Ψ_6-autocorrelation time and the mixing time.

3. Markov chains and lifting

In this section the basic concepts of Markov-chain Monte Carlo and lifting are introduced, following [19, 21].

Hard disks have a continuous high-dimensional configuration space. For the discussion of Markov-chain Monte Carlo and lifting, consider a toy-model system with a finite number of $L \in \mathbb{N}$ states on a circle. Let each state i have the stationary weight $\pi_i > 0$. The aim of MCMC is to sample the states according to the target distribution $\underline{\pi} = (\pi_1, \dots, \pi_L)$. To this end a MCMC algorithm advances from state i to state j with probability T_{ij} in a move. These probabilities are the elements of the algorithm's transition matrix T, which represents the algorithm. After t moves the Markov chain is in state i with probability $\varpi_i(t)$. The probability distribution $\underline{\varpi}(t) = (\varpi_1(t), \dots, \varpi_L(t))$ after t moves depends only on the distribution that resulted from the previous move and on the transition probabilities. This is expressed in the master equation

$$\underline{\varpi}(t+1) = T\underline{\varpi}(t), \tag{3}$$

hence $\underline{\varpi}(t)$ depends only on T and $\underline{\varpi}(0)$,

$$\underline{\varpi}(t) = T^t\underline{\varpi}(0). \tag{4}$$

To guarantee convergence $\underline{\varpi}(t) \to \underline{\pi}$ for $t \to \infty$, the transition matrix must fulfill three conditions: First it must be *aperiodic*, meaning that the Markov chain be free of cycles. Second it must be *irreducible*, to ensure that $\underline{\pi}$ is unique. Third, to ensure convergence, the transition matrix must fulfill the *global balance condition*

$$\underline{\pi} = T\underline{\pi}. \tag{5}$$

A *reversible* Markov chain is one that fulfills the stronger *detailed balance condition*

$$\pi_i T_{ij} = \pi_j T_{ji} \tag{6}$$

For example, for $\pi_i = 1/L$ for all i, a simple reversible algorithm is a random walk on the circle: From state i with the holding probability p it stays at i and with probability $q = (1-p)/2$ it advances to state $i+1$, or $i-1$ respectively. The corresponding transition matrix for $L = 4$ is

$$T = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{pmatrix} p & q & & q \\ q & p & q & \\ & q & p & q \\ q & & q & p \end{pmatrix} \end{array}. \tag{7}$$

MCMC algorithms not fulfilling detailed balance are called *irreversible*.

The rate of convergence of $\underline{\varpi}(t)$ towards $\underline{\pi}$ is determined by the eigenvalues of T. For aperiodic, irreducible Markov chains the largest eigenvalue of the transition matrix is 1 with multiplicity one. The other eigenvalues may be complex if the algorithm represented by T is irreversible, but still

$$1 = \lambda_1 > |\lambda_2| \geq \dots \geq |\lambda_L| \geq 0 \tag{8}$$

holds. The probability distribution $\underline{\varpi}(t)$ can be expanded in the eigenbasis of T. The eigenvector corresponding to λ_1 is $\underline{\pi}$. Evidently, by repeatedly multiplying the current probability distribution with T, the contributions of all eigenvectors but $\underline{\pi}$ decay exponentially. The convergence rate of the algorithm is determined by the second-largest eigenvalue λ_2, such that the mixing time – the timescale after which the Markov chain is in equilibrium is given by

$$\tau_{\text{mix}} = -\frac{1}{\ln|\lambda_2|}. \tag{9}$$

Therefore the smaller $|\lambda_2|$, the faster does the algorithm converge. Convergence in terms of an observable does not necessarily depend on all eigenvalues. Hence different observables converge with different speed.

Lifting a Markov-chain algorithm means to augment the configuration space with an auxiliary variable, the *lifting variable*. The stationary probabilities and observables on the original configuration space are obtained by averaging out the lifting variable. Lifting and violating detailed balance can speed up convergence, but it does not always. For discrete Markov chains it is shown that in the optimal speedup, the mixing time goes over to its square root [20]. In the exemplary random walk the lifting variable is ε, which takes values in ± 1 and denotes the direction the random walk advances. The lifted transition matrix is

$$T = \begin{array}{c} \\ (1,+) \\ (2,+) \\ (3,+) \\ (4,+) \\ (1,-) \\ (2,-) \\ (3,-) \\ (4,-) \end{array} \begin{array}{c} \begin{array}{cccccccc} (1,+) & (2,+) & (3,+) & (4,+) & (1,-) & (2,-) & (3,-) & (4,-) \end{array} \\ \begin{pmatrix} p_\text{h} & q & & & p_\text{l} & & & \\ & p_\text{h} & q & & & p_\text{l} & & \\ & & p_\text{h} & q & & & p_\text{l} & \\ q & & & p_\text{h} & & & & p_\text{l} \\ p_\text{l} & & & & p_\text{h} & & & q \\ & p_\text{l} & & & q & p_\text{h} & & \\ & & p_\text{l} & & & q & p_\text{h} & \\ & & & p_\text{l} & & & q & p_\text{h} \end{pmatrix} \end{array}. \tag{10}$$

In each move the lifted algorithm either stays in state (i, ε) with the holding probability p_h or flips ε with the lifting probability p_l or advances to state $(i + \varepsilon, \varepsilon)$ with probability $q = 1 - p_\text{h} - p_\text{l}$. In a *lifting move* the lifting variable changes. For simplicity we choose $p_\text{h} = p_\text{l} = p$.

In the unlifted Eq. (7), as well as in the lifted case Eq. (10), the eigenvalues of T can be computed numerically for a given p. With Nelder-Mead minimization of $|\lambda_2|$ as a function of p the optimal value of p and the resulting minimal $|\lambda_2|$ is determined for a given L. As demonstrated in Fig. 2 the mixing time is reduced to approximately its square root. For the lifted random walk an intuitive assumption is that the optimal p has the property, that $\mathcal{O}(L)$ states are visited between two lifting moves. Then the states are sampled in a coherent fashion. Indeed this is the optimal scaling of p. The probability for l displacement moves in succession, followed by a lifting move is $(1 - 2p)^l p$, ignoring any holding moves in between. Consequently the mean length of persistent motion in

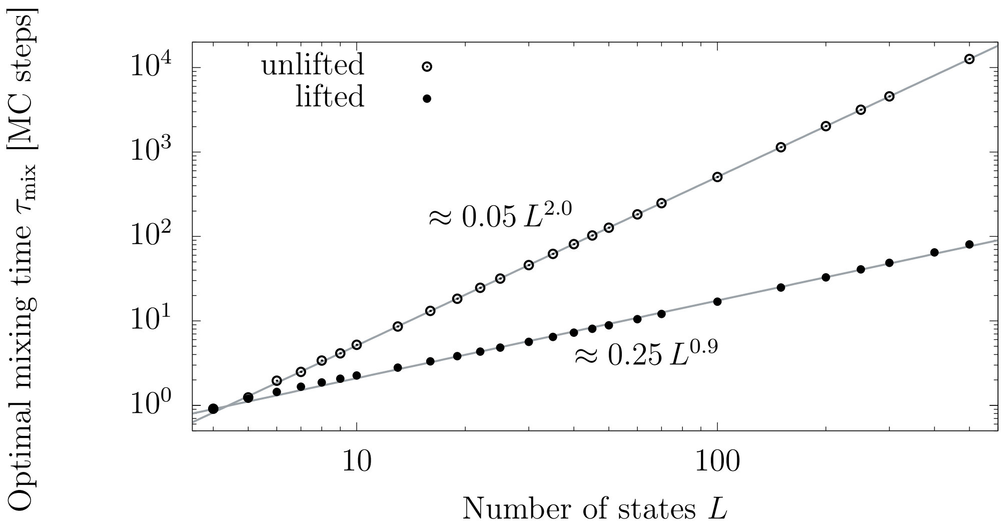


Figure 2: Minimal mixing times of the unlifted Eq. (7) and lifted Eq. (10) random walk on a circle. The mixing times have been minimized by numerical determination of the optimal value for the parameter p. By lifting, the mixing time reduces here to its square root.

one direction is

$$\langle l \rangle = \sum_{l=0}^{\infty} l(1-2p)^l p = \frac{1-2p}{(2p)^2} p = \frac{1}{4p} - \frac{1}{2} \tag{11}$$

Solving for p yields

$$p = \frac{1}{4\langle l \rangle + 2}. \tag{12}$$

This relation fits the numerically optimized p as a function of L, with $\langle l \rangle = 0.04L$. Although the prefactor of four percent is surprisingly small, the length of persistent motion is proportional to L for optimal mixing.

4. Monte Carlo algorithms for hard disks

For hard disks the stationary weight of a configuration vanishes, if the configuration contains overlapping particles, and is a finite constant otherwise. Indeed this is expressed by the Boltzmann factor, with the internal energy being the sum of all hard-particle pair potentials. In $D > 1$ direct sampling from the stationary distribution is virtually only possible in the very low density regime, see [22], chapter 2.2.1. For this reason, when studying the hard-disk melting transitions, other sampling methods, like MCMC, must be employed.

4.1. Metropolis Monte Carlo (MMC)

We use the Metropolis Monte Carlo algorithm (MMC, [8]) as a reference method for consistency checks to show our new algorithms be correct and for performance comparison. By fulfilling detailed balance MMC fulfills global balance automatically. The single particle moves are rejected if an overlap would be generated and the moves are incoherent, as there is no coordination between them. Therefore at high packing fractions MMC is slow, as the particles restrain each other. One MMC step for hard particles comprises:

1. Choose a random particle i, with i uniformly distributed in $\{1, \dots, N\}$
2. Propose a new position for this particle

$$\vec{\mathbf{r}}_i' = \vec{\mathbf{r}}_i + \sigma \vec{\mathbf{R}}, \tag{13}$$

   with a vector $\vec{\mathbf{R}}$, randomly sampled in each MMC step.
3. Accept the move if it produces no overlap; otherwise reject it, *i. e.* keep the old values of $\vec{\mathbf{r}}_i$ and $\vec{\mathbf{c}}_i$.
4. In every s^{th} MMC step: Take values for averaging observables.

The random vector $\vec{\mathbf{R}}$ in step 2 is sampled from the uniform distribution on the interval $[-R_{\text{MMC}}/2, R_{\text{MMC}}/2)$, with the dimensionless relative move distance $R_{\text{MMC}} > 0$. Although anisotropic, this is a sufficient choice for the distribution of $\vec{\mathbf{R}}$, since mutually reverse moves are proposed with equal probability. In other words, the a-priori probability of proposing a move from configuration a to configuration b is symmetric in a and b, which is required for detailed balance in the Metropolis algorithm, see [22], chapter 1.1.6[1].

The overlap test in step 3 is efficiently implemented using a cell subdivision (for details see appendix A.2). For soft particles step 3 would be to accept the move with an acceptance probability depending on the temperature and the energy difference between the new and old configuration.

[1]Note that in polydisperse systems it may also be beneficial to propose shorter displacements for larger particles, in order to address the 'pope in the crowd' effect [22], chapter 6.1.3.

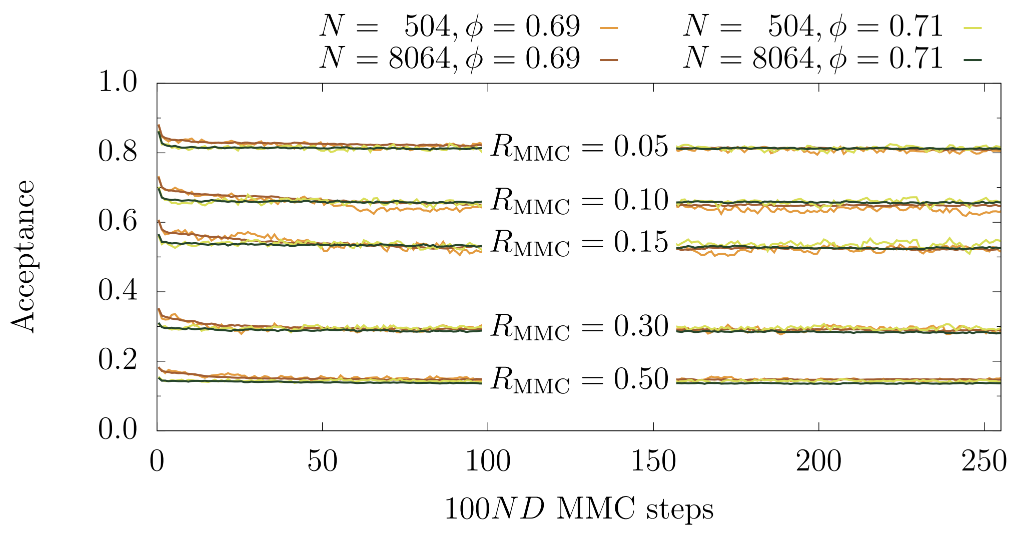


Figure 3: Acceptance of MMC over time, for different move distances R_{MMC} and at different packing fractions ϕ and particle numbers N. Particles are initialized on a triangular lattice. Acceptances are averaged over $100ND$ proposed moves.

Sampling in step 4 takes place every s MMC steps, irrespective of the number of rejections. Configurations do not undergo large changes within few MMC steps; and evaluating the observables for a sample can be a costly operation (depending on the observables). Thus speed is easily gained by choosing $s \sim N$.

The relative move distance R_{MMC} is a tunable simulation parameter with no physical meaning. It controls the *acceptance*, the ratio of accepted to proposed moves, and thus the convergence towards equilibrium. Low acceptance is found when R_{MMC} is chosen too large. Then most moves produce overlap and must be rejected – a waste of computation time. On the other hand high acceptance, associated with small R_{MMC}, indicates that computation time is spent on many consecutive short moves, which could be replaced by few longer moves, increasing the efficiency of the simulation. Optimal acceptance lies, as a rule of thumb, on the order of 1/2 [22], chapter 1.1.2. We choose $R_{\mathrm{MMC}} = 0.15$ to fall into this regime, see Fig. 3.

4.2. Event-chain Monte Carlo (ECMC)

ECMC, as described in [17, 18], is a lifted Markov-chain algorithm, with the lifting variables $l \in \{1, \dots, N\}$, labelling the particle that is currently displaced, and $\vec{\mathbf{v}} \in \{+\vec{\mathbf{e}}_1, +\vec{\mathbf{e}}_2\}$, the direction of displacement. It is rejection-free and fulfills maximal global balance, *i. e.* it is irreversible. If a move $a \to b$ is performed, the reverse move $b \to a$ is never proposed. Instead the stationary flow into configuration a returns via PBC.

The basic move is an infinitesimal displacement of particle $l = i$ by $\mathrm{d}\vec{\mathbf{r}} = \vec{\mathbf{v}}\mathrm{d}t$. The particle i is successively displaced, until it collides with another particle j. In practical implementations infinitesimal moves are realized based on events – particle i is directly displaced from its initial position to the position of the collision event. For details on predicting, scheduling, and processing events, see appendix A.3; our implementation of ECMC also makes use of a cell subdivision for efficiency, see appendix A.2. The *collision event* is a lifting event, upon which particle i stops and particle j moves with $\vec{\mathbf{v}}$ – the lifting variable l changes. This goes on for the *chain duration* t_{ch}, or equivalently until the sum of particle displacements equals the *chain displacement* $\ell_{\mathrm{ch}} = \|\vec{\mathbf{v}}\|\, t_{\mathrm{ch}}$. Then a different type of lifting event, the *chain-end event*, completes the event chain. The chain duration (or chain displacement) is the simulation parameter of ECMC. It can be sampled from a distribution for every chain or, as in our case, be constant throughout the simulation. For the next event chain l is resampled from $\{1, \dots, N\}$ and $\vec{\mathbf{v}}$ is redrawn from $\{+\vec{\mathbf{e}}_1, +\vec{\mathbf{e}}_2\}$ sequentially.

In a single chain all displacements advance the particles in the same direction. But due to the atomisitc nature of the particle-based system the chain trajectories are diffusive in the directions normal to $\vec{\mathbf{v}}$, as can be seen in the top row on page 3. Although ECMC is a local algorithm, because its elementary moves are the infinitesimal one-particle displacements, it displaces particles coherently.

It is not correct to take a sample after every s^{th} lifting event. This would have the consequence that almost every sample (except those coinciding by chance with chain-end events) would contain a pair of disks in contact, which actually occurs only with vanishing probability. To avoid such biased sampling, samples must be taken at events, which are not related to collision events. It is correct to sample after an arbitrary fixed or somehow distributed time interval, introducing sampling events. However for simplicity, we choose the special case, where those sampling events are identical with every s^{th} chain end.

The *total chain displacement* is the length of the chain trajectory depicted on page 3 in direction $\vec{\mathbf{v}}$, measured across the periodic boundary. It exceeds the chain displacement by the fluctuating *excess displacement* ℓ_{ex}. When the two particles i and j collide, the distance along $\vec{\mathbf{v}}$ in between is not travelled by a particle. Still this distance, which is less or equals $\sigma/2$, contributes to the total displacement. As outlined in section 4.2.1, knowledge of the total displacement permits the calculation of pressure. One might ask if prescribing a fixed total displacement, instead of the chain displacement, is valid as well. This is not possible, because many chains would end in a conflict: As long as the demanded total displacement is not yet accumulated, another collision must occur, but the next collision could add already too much excess displacement. Then the resulting total displacement would inevitably be either too small or too large. Similarly, it is not possible to choose a number of collisions per chain, because it would be unclear how far the last particle in a chain should be displaced. Also such moves cannot be derived from the picture of infinitesimal moves.

Although implementations of ECMC are event-based, ECMC is not a molecular dynamics (MD) simulation. Newtonian collision dynamics are fully determined by the conservation of momentum and energy and the fact that a force (change of velocity) is directed along the bond vector at collision, see [23], chapter 3.6.1 and [24]. The global balance condition is respected by MD as well. But MD also mixes the particle velocities.

4.2.1. Calculation of pressure

The *reduced pressure* (or compressibility factor) is defined as

$$Z := \frac{\beta P}{\rho} \tag{14}$$

with inverse temperature β, pressure P, and particle number density $\rho = N/V_{\mathbb{B}} = 4\phi/\left(\pi\sigma^2\right)$. In hard particle systems the temperature depends only on the (physical) kinetic energy of the particles, which is trivially factored out of Monte Carlo simulations. The pair potential cannot define a temperature scale, thus we choose $\beta = 1$ in natural units. For straight constant-speed ECMC (constant $\vec{\mathbf{v}}$ throughout a chain, $\|\vec{\mathbf{v}}\| = 1$), [18] derives the identity

$$Z = 1 + \frac{\langle \ell_{\text{ex}} \rangle_{\text{chains}}}{\ell_{\text{ch}}}, \tag{15}$$

where the first summand, 1, constitutes the ideal gas pressure, while the second summand amounts to the excess pressure. It contains the chain displacement ℓ_{ch} and the excess displacement ℓ_{ex}. By definition a straight chain's total displacement $\ell_{\text{ch}} + \ell_{\text{ex}}$ is the projection of $\vec{\mathbf{r}}_f - \vec{\mathbf{r}}_i$ on the direction of $\vec{\mathbf{v}}$, with $\vec{\mathbf{r}}_i$ the initial position of the first particle in the chain and $\vec{\mathbf{r}}_f$ the final position of the last particle. The total displacement is larger than ℓ_{ch} and may also be larger than the periods of the simulation box $\mathbb{B}$. The excess displacement is the portion of total displacement, that does not arise from particle displacements but from collisions,

$$\ell_{\text{ex}} = \left(\sum_{\text{collisions}} \vec{\mathbf{r}}_{ij} \right) \cdot \vec{\mathbf{v}} / \|\vec{\mathbf{v}}\|, \tag{16}$$

where $\vec{\mathbf{r}}_{ij}$ is the bond vector of particles i and j at the instance of collision. Summands fulfill the inequalities $0 \leq \vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}} / \|\vec{\mathbf{v}}\| \leq \sigma$ where the extremal values correspond to lateral and frontal (head-on) collision.

Eq. (15) was shown in [18] for straight ECMC. More general chains with variable direction or even magnitude of $\vec{\mathbf{v}}$ were not considered so far. Here we generalize Eq. (15) to arbitrary variable chain velocity $\vec{\mathbf{v}}$. First consider the direction of $\vec{\mathbf{v}}$ varying with time during the chain. Since the chains are no longer straight, the above definition of excess displacement must be generalized: $\vec{\mathbf{v}} / \|\vec{\mathbf{v}}\|$ cannot be factored out of the summation in Eq. (16) anymore. As long as the magnitude of velocity is constant and only the direction varies, displacements and durations are directly proportional.

In the case of a varying magnitude of velocity, $\|\vec{\mathbf{v}}\| \neq$ const., even ℓ_{ch} is no longer a constant simulation parameter; it becomes fluctuating instead. Hence it is convenient to formulate a generalization of Eq. (15) in terms of durations. The role of the excess displacement is now played by the *excess duration*

$$t_{\text{ex}} = \sum_{\text{collisions}} \left(\vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}} / \|\vec{\mathbf{v}}\|^2 \right). \tag{17}$$

Here the velocity may vary from one collision to another and thus cannot be factored out. Still single summands are always positive and less or equal $\sigma/\|\vec{\mathbf{v}}\|$. With t_{ex} we

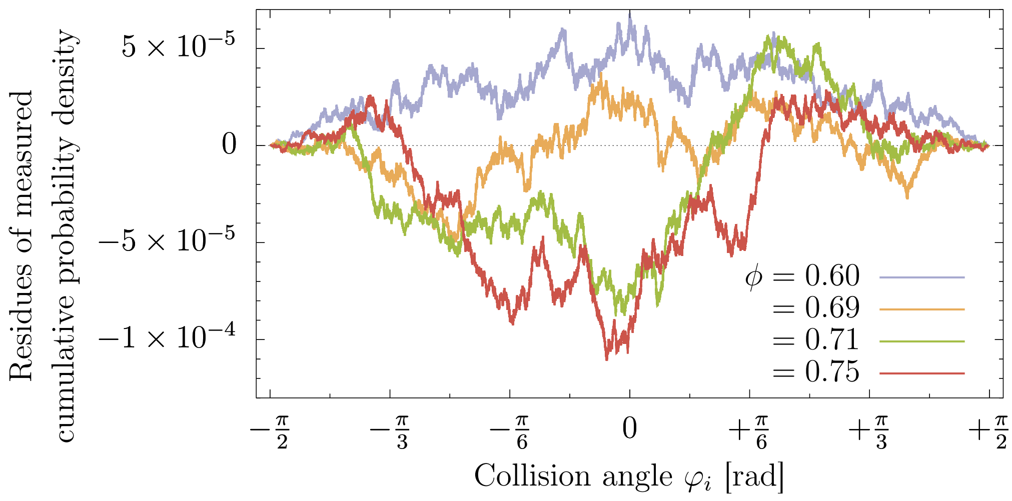


Figure 4: The cumulative distribution of the collision angle is measured with $\vec{\mathbf{v}}$ sampling all directions in the plane ($> 10^8$ samples). From Eq. (19) it follows that the cumulative distribution is $\frac{1}{2}\sin(\varphi_i)$. Subtracting this from the measured cumulative distribution reveals unstructured residues, independent of packing fraction.

obtain

$$Z = 1 + \frac{\langle t_{\mathrm{ex}} \rangle_{\mathrm{chains}}}{t_{\mathrm{ch}}}. \tag{18}$$

4.2.2. Distribution of the collision angle and the mean number of collisions per chain

The distribution of collision angles in ECMC has not been reported on in the literature. Let the collision angle φ_i be defined as the angle between $\vec{\mathbf{v}}$ and the bond vector $\vec{\mathbf{r}}_{ij}$ at the instance of collision. The probability density distribution of φ_i, is the conditional probability density distribution of observing a collision angle φ_i at a collision event E:

$$w(\varphi_i) = P(\varphi_i|E) = \frac{P(\varphi_i \cap E)}{P(E)} = \frac{p_{\mathrm{lift}}}{\int_{-\pi/2}^{\pi/2} p_{\mathrm{lift}}\ \mathrm{d}\varphi_i} = \frac{\cos\varphi_i}{\int_{-\pi/2}^{\pi/2} \cos\varphi_i\ \mathrm{d}\varphi_i} = \frac{1}{2}\cos\varphi_i, \tag{19}$$

with the lifting probability as in [18] being either zero, if $\mathrm{d}U \leq 0$, or otherwise $p_{\mathrm{lift}} = \beta \mathrm{d}U = \beta \frac{\partial U}{\partial r}\frac{\partial r}{\partial x}\mathrm{d}x$ and $\frac{\partial r}{\partial x} = \cos\varphi_i$. When φ_i is sampled, with $\vec{\mathbf{v}}$ uniformly distributed over all directions, φ_i is indeed found to be distributed with $w(\varphi_i)$. Fig. 4 shows perfect agreement with numerical data. As this property is purely geometric, it is independent of packing fraction and does even hold in the anisotropic solid phase.

Knowing the distribution $w(\varphi_i)$, one can for example derive the exact relation between the mean number of collisions per chain and the excess displacement. For a quantity q, let $\{q\}_{\varphi_i}$ denote its average over collision angle φ_i,

$$\{q\}_{\varphi_i} := \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} q(\varphi_i) w(\varphi_i) \mathrm{d}\varphi_i = \frac{1}{2} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} q(\varphi_i) \cos\varphi_i \mathrm{d}\varphi_i. \tag{20}$$

A collision at angle φ_i contributes $\sigma\cos\varphi_i$ to ℓ_{ex}, on average

$$\{\sigma\cos\varphi_i\}_{\varphi_i} = \sigma \int_{-\pi/2}^{\pi/2} w(\varphi_i) \cos\varphi_i \mathrm{d}\varphi_i = \frac{1}{2}\sigma \int_{-\pi/2}^{\pi/2} \cos^2\varphi_i \mathrm{d}\varphi_i = \frac{\pi}{4}\sigma. \tag{21}$$

Hence the mean number of collisions per chain is, using Eq. (15),

$$\frac{4\left\langle \ell_{\mathrm{ex}} \right\rangle_{\mathrm{chains}}}{\pi\sigma} = \frac{4\ell_{\mathrm{ch}}}{\pi\sigma}\left(\frac{\left\langle \ell_{\mathrm{ex}} \right\rangle_{\mathrm{chains}}}{\ell_{\mathrm{ch}}} + 1 - 1\right) = \frac{4\ell_{\mathrm{ch}}}{\pi\sigma}(Z-1). \tag{22}$$

Of course it depends linearly on the chain displacement ℓ_{ch} and is packing fraction dependent via the pressure Z. In the ideal gas limit, when $Z \searrow 1$, the number of collisions per chain tends to zero.

4.2.3. Compact initialization as a degenerate case

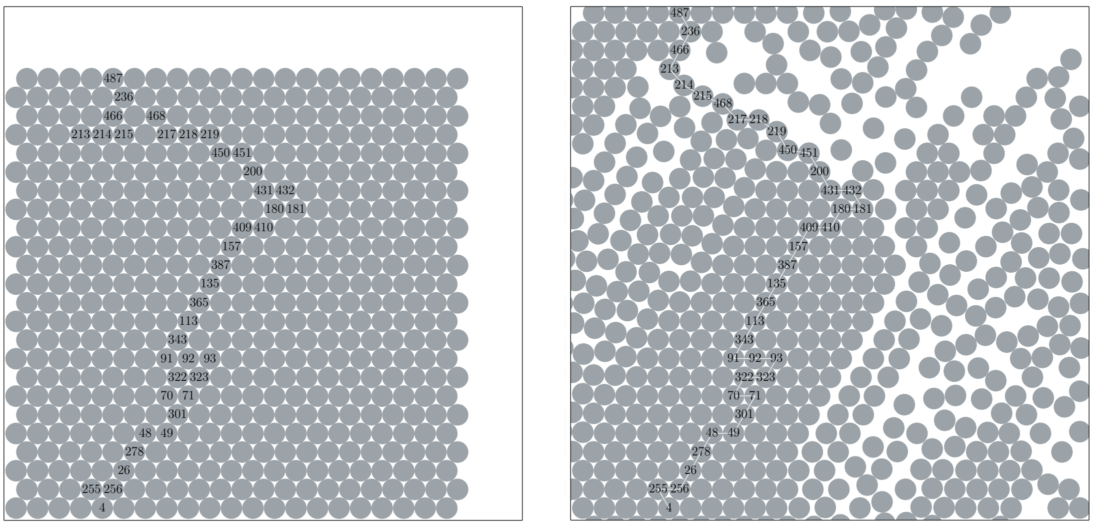
ECMC may fail to converge to equilibrium, if it is performed on an initial configuration which contains particles touching several other particles. The fundamental problem is that the initial configuration contains particles which will collide with several other particles simultaneously. Then the lifting move is ambiguous – either of the hit particles could be the new moving particle. As there are several (lifted) configurations tracing back to the same prior configuration, the global balance condition is violated. This issue is not resolved in the framework of ECMC. Rather it is argued in [18], that infinitesimal moves ensure that there are no simultaneous collisions. This is true for typical equilibrium configurations, while the pathological cases, in which simultaneous collisions occur, are only a set of measure zero in the configuration space. But here we choose one configuration from this set as initial configuration. In order to ensure convergence, a possible modification of ECMC would be to randomly choose one of the candidates for lifting. However this is numerically unfeasible, because it is not possible to reliably identify simultaneous collision events in floating-point arithmetic. Instead the ambiguity is resolved with a bias in our implementation, depending on the scheduling of simultaneous events. Consequently the distribution of the lifting variable is not guaranteed to be stationary.

In this situation ECMC may form a system-spanning line of touching particles. As the particles are in contact, they cannot be displaced and collisions between them are *instantaneous*. Therefore collisions do not contribute to the chain displacement, or chain duration respectively. As a consequence no other events take place in between – no cell crossings and in particular no chain end. Since an event chain cannot end with an instantaneous collision, a system-spanning line of touching disks can trap the ECMC algorithm in an infinite chain. Detecting such an infinite chain is algorithmically not straight-forward. Let the numbers identify particles involved in a collision event and let the asterisk * denote an instantaneous collision. Then an infinite chain can be of the schematic form

$$1 \to 2 \to 3 \overset{*}{\to} 4 \to 5 \overset{*}{\to} 6 \overset{*}{\to} 7 \overset{*}{\to} 8 \overset{*}{\to} \underbrace{6 \overset{*}{\to} 7 \overset{*}{\to} 8 \overset{*}{\to}}_{\text{cyclic}} \cdots \tag{23}$$

Hence checking for the recurrences of events $1 \to 2$, the beginning of the chain, $3 \overset{*}{\to} 4$, the first instantaneous collision, or $5 \overset{*}{\to} 6$, the first in the row of successive instantaneous collisions, does not provide secure means to detect an infinite chain. Instead we monitor if more than N instantaneous collisions occur in succession. If this happens, it is clear that either all or a subset of the particles take part in an infinite chain. Then we abort the chain. The subsequent event chains dissolve the system-spanning line of touching particles. Although aborting an unfinished event chain is forbidden by the concept of ECMC, in this case virtually no error is made, as the respective configurations belong to a set of measure zero in the configuration space.

In principle this problem exists for the regular lattice initial configuration (see Fig. 17a and appendix A.1); yet there infinite chains do not emerge. When starting from the dilute regular lattice, collisions are not instantaneous in the first place, *i. e.* particles are displaced, resolving the ambiguities early in the simulation. But when a compact crystal (see Fig. 5a and appendix A.1) is chosen as initial condition, the probability of forming an infinite chain is rather high. In the bulk of the compact crystal all collisions are instantaneous. When the event chain eventually reaches the surface the pending displacement does not depend on the number of events already occurred in the chain. Thus two chains starting form different places in the bulk can produce the same outcome. If the lattice orientation, the box orientation, and the direction of the straight event chains are aligned – which is true in our setup – and the chain displacement ℓ_{ch} is long enough, the following will happen: Assume the chain starts somewhere in the bulk. After a number of instantaneous collisions the first surface particle is reached. It is displaced across the periodic box and attached to the lattice again on the other side. The chain continues until the demanded chain displacement is accomplished with the final displacement of a surface particle to a position somewhere in the void. Eventually a line of contacts is formed as in Fig. 5b. From compact initialization, infinite chains are observed in systems with moderate numbers of particles (up to $N \approx 5000$) in about one tenth of the simulation runs. In larger systems the phenomenon is suppressed. In future this problem should be avoided by choosing only nearly compact initial configurations with random noise on the particle positions.



(a) initially (b) during 75th event chain

Figure 5: Straight ECMC with a constant chain displacement $\ell_{\text{ch}} = 0.1 L_{\mathbb{B}}$ can create (b) a system-spanning line of contacts from (a) the compact initial configuration for $N = 504$, $\phi = 0.69$. Remarkably, not all particles in the line of contacts lie on lattice sites. The particles eventually involved in the infinite chain are labelled by numbers in both pictures. An event chain in direction $\vec{\mathbf{e}}_2$, involving the labelled particles, only consists of instantaneous collisions and cyclically follows the trajectory marked as a white line in (b). The horizontal segments in the chain are determined by the sequence by which the particles are registered in the cell subdivision and the sequence neighboring cells are inspected.

5. Design considerations of arcuate ECMC for hard disks

For setting up arcuate event chains in $D = 2$ dimensions we follow two approaches: First altering the direction of movement $\vec{\mathbf{v}}$ at each collision and second altering $\vec{\mathbf{v}}$ at new lifting events independent of collisions. The first approach results in a valid but unfeasible algorithm, turn-at-collision-ECMC. Using the second approach SquareMC and the more general ArcMC are constructed.

In principle it would be possible to extend the algorithms presented here to higher dimensions $D > 2$, by in-plane rotation of $\vec{\mathbf{v}}$. The planes to which $\vec{\mathbf{v}}$ would be confined could be the $\binom{D}{2} = D(D-1)/2$ independent planes, chosen from sequentially, or arbitrary (random) planes. However it is disputable whether rotations are what ECMC needs in $D > 2$.

5.1. Arcuate ECMC turning at collisions, with variable magnitude of velocity

Introducing a velocity $\vec{\mathbf{v}}$, with which the lifted particle moves, seems unphysical and only useful for building up analogy to MD. There is no thermal velocity defined in ECMC which would provide a scale for $\vec{\mathbf{v}}$. Hence it seems that a velocity could always be factored out and dynamics could be expressed in terms of distances rather than times, like in [18]. However when, as a generalization of ECMC, the direction of movement is allowed to change at each collision and not only from chain to chain, then the global balance condition requires also the magnitude of the velocity to change. The notion of time is introduced via the Markov-chain formalism. One must respect that the proportionality factor connecting $\mathrm{d}\vec{\mathbf{r}}$ and $\mathrm{d}t$ is not constant when the direction of motion changes.

The stationary weight $\pi(a)$ of a physical configuration a is up to normalization the Boltzmann factor $\exp(-\beta U(a))$, where $U(a)$ is the energy of the configuration. Here $U(a)$ is treated as in the case of soft particles, where derivatives exist. The results also apply to hard particles as a limit. A lifted configuration a_j has the stationary weight

$$\pi(a_j) = \pi(b_j)p_{\mathrm{acc}}(j) + \pi(a_i)\left(1 - p_{\mathrm{acc}}(i)\right). \tag{24}$$

The configurations b_j and a_i are the possible priors of a_j: b_j is the configuration before particle j has performed an infinitesimal move, *i. e.* compared with a_j particle j is moved backward by $\mathrm{d}\vec{\mathbf{r}}_j = \vec{\mathbf{v}}_j\mathrm{d}t$. a_i is the configuration before a lifting move $i \to j$ has occurred, *i. e.* a_i is the same physical configuration as a_j but with lifted particle i. The acceptance probability $p_{\mathrm{acc}}(j)$ for the move of particle j can safely be assumed to be is equal to unity. This is so because either the configuration b_j is not valid, then $\pi(b_j)$ vanishes anyway, or b_j is valid and the move $b_j \to a_j$ has already occurred. The term $1 - p_{\mathrm{acc}}(i)$ gives the lifting probability. To first order in $\mathrm{d}\vec{\mathbf{r}}_i$, $\mathrm{d}\vec{\mathbf{r}}_j$ this equation is equivalent to

$$\exp(-\beta U) = \exp\left(-\beta U + \beta\frac{\partial U}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_j}\cdot\vec{\mathbf{v}}_j\mathrm{d}t\right) + \exp(-\beta U)\left(1 - \exp\left(-\beta\frac{\partial U}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_i}\cdot\vec{\mathbf{v}}_i\mathrm{d}t\right)\right) \tag{25}$$

with the bond vector $\vec{\mathbf{r}}_{ij} := \vec{\mathbf{r}}_j - \vec{\mathbf{r}}_i$, $r_{ij} \equiv \|\vec{\mathbf{r}}_{ij}\|$, and $U \equiv U(r_{ij})$, as only the pair potential of particles i and j contributes. The term $\exp(-\beta U)$ is factored out to get

$$1 = \exp\left(\beta \frac{\partial U}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_j} \cdot \vec{\mathbf{v}}_j \mathrm{d}t\right) + 1 - \exp\left(-\beta \frac{\partial U}{\partial r_{ij}} \frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_i} \cdot \vec{\mathbf{v}}_i \mathrm{d}t\right). \tag{26}$$

Consequently the arguments of the exponential functions must equal, therefore

$$\frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_j} \cdot \vec{\mathbf{v}}_j \mathrm{d}t = -\frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_i} \cdot \vec{\mathbf{v}}_i \mathrm{d}t. \tag{27}$$

Since $\frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_j} = -\frac{\partial r_{ij}}{\partial \vec{\mathbf{r}}_i} = \frac{\vec{\mathbf{r}}_{ij}}{r_{ij}}$, the global balance condition is

$$\vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}}_i = \vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}}_j. \tag{28}$$

This means that the velocity component parallel to the bond vector $\vec{\mathbf{r}}_{ij}$ (at the instance of collision) is conserved in a collision. Indeed this does also hold for Newtonian dynamics, where velocity components parallel to the bond vector are exchanged upon collision. There Eq. (28) is given in the form $\vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}}_i = \vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}}_j{}'$, with $\vec{\mathbf{v}}_i$ before, $\vec{\mathbf{r}}_{ij}$ at, and $\vec{\mathbf{v}}_j{}'$ after the collision. Yet Newtonian dynamics leave the normal components of velocities unchanged. Therefore in MD, starting with one moving particle and another one at rest, after the collision there will be two moving particles (except for head-on collisions).

In ECMC the velocity of particle i must be zero after collision. The velocity of particle j (after collision) must fulfill the global balance condition, Eq. (28), which is a condition on the component parallel to the bond vector. The normal component can be altered arbitrarily without violating global balance, as long as the mapping $\vec{\mathbf{v}}_i \mapsto \vec{\mathbf{v}}_j$ is injective. Otherwise there exists no unique inverse map $\vec{\mathbf{v}}_i(\vec{\mathbf{v}}_j)$, which is needed to unambiguously identify the priors of configuration a_j. In the standard ECMC algorithm [18] Eq. (28) is simply fulfilled via $\vec{\mathbf{v}}_i = \vec{\mathbf{v}}_j$. But projection conservation is also respected in the *reflected event-chain algorithm* presented in [17], where the normal component of the velocity is reflected in each move, resulting in the chains "meander[ing] through the system" [17] and thus reducing performance.

5.1.1. Impossibility of imposing an angle

In order to construct arcuate chains in $D = 2$ dimensions, it stands to reason to specify an angle θ between incoming and outgoing velocity. However this is not possible. Considering the angles φ_i, φ_j between $\vec{\mathbf{r}}_{ij}$ and $\vec{\mathbf{v}}_i, \vec{\mathbf{v}}_j$ and demanding $\varphi_j - \varphi_i = \theta$, Eq. (28) yields

$$r_{ij}\, \|\vec{\mathbf{v}}_i\| \cos\varphi_i = r_{ij}\, \|\vec{\mathbf{v}}_j\| \cos\varphi_j \qquad \Rightarrow \qquad \|\vec{\mathbf{v}}_j\| = \|\vec{\mathbf{v}}_i\| \frac{\cos\varphi_i}{\cos\varphi_j} = \|\vec{\mathbf{v}}_i\| \frac{\cos\varphi_i}{\cos(\theta + \varphi_i)}. \tag{29}$$

For any value of θ, the fraction of cosines may in general be negative (depending on φ_i), implying a negative norm. This just expresses the fact that an angle θ cannot be employed, if the resulting φ_j would lie outside of the range $\left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$, because then $\vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}}_j$ would have the opposite sign of $\vec{\mathbf{r}}_{ij} \cdot \vec{\mathbf{v}}_i$. Moreover $\varphi_j \notin \left[-\frac{\pi}{2}, +\frac{\pi}{2}\right]$ implies that particle j would move into particle i, instead of moving away from it.

5.1.2. ECMC turning at collisions (turn-at-collision-ECMC)

Since prescribing an angle between incoming and outgoing velocity is in general not possible, the next idea is to always add a normal velocity:

$$\vec{\mathbf{v}}_j = \vec{\mathbf{v}}_i + \Delta v \vec{\mathbf{e}}_\perp, \tag{30}$$

with a function $\Delta v(\|\vec{\mathbf{v}}_i\|, \varphi_i)$ and the normal vector $\vec{\mathbf{e}}_\perp \perp \vec{\mathbf{r}}_{ij}$. The (extraphysical) kinetic energy after collision is thus

$$\|\vec{\mathbf{v}}_j\|^2 = \|\vec{\mathbf{v}}_i\|^2 + \Delta v^2 + 2\|\vec{\mathbf{v}}_i\| \Delta v \sin\varphi_i \tag{31}$$

or as a ratio

$$\frac{\|\vec{\mathbf{v}}_j\|^2}{\|\vec{\mathbf{v}}_i\|^2} = 1 + \left(\frac{\Delta v}{\|\vec{\mathbf{v}}_i\|}\right)^2 + 2\frac{\Delta v}{\|\vec{\mathbf{v}}_i\|} \sin\varphi_i. \tag{32}$$

The normal vector is chosen such that the addition systematically rotates the direction of motion. In $D = 2$ this is easily achieved by setting

$$\vec{\mathbf{e}}_\perp(\vec{\mathbf{r}}_{ij}) = \begin{pmatrix} -(\vec{\mathbf{r}}_{ij})_2 \\ (\vec{\mathbf{r}}_{ij})_1 \end{pmatrix}, \tag{33}$$

which will enforce a counter-clockwise rotation. For Δv as a function of $\|\vec{\mathbf{v}}_i\|$ and φ_i consider the four options

$$0, \tag{34}$$
$$-2\|\vec{\mathbf{v}}_i\| \sin\varphi_i, \tag{35}$$
$$\text{const.}, \tag{36}$$
$$-\|\vec{\mathbf{v}}_i\| \sin\varphi_i + \text{const.} \tag{37}$$

In the following we will demonstrate that two of the four choices represent algorithms previously known, one yields a new, valid, but not viable algorithm and one is not an allowed choice.

Obviously Eq. (34) is ECMC. The term $\|\vec{\mathbf{v}}_i\| \sin\varphi_i \vec{\mathbf{e}}_\perp$ is the normal component of $\vec{\mathbf{v}}_i$. Subtracting it twice from $\vec{\mathbf{v}}_i$ is equivalent to a reflection. Hence Eq. (35) represents the reflected event-chain algorithm. Those two choices are the only ones, that exactly conserve the kinetic energy in each collision, while respecting global balance. Yet both algorithms do not exhibit arcuate chains. Therefore in order to construct a new algorithm the exact conservation of kinetic energy must be dropped. The variable kinetic energy is either only statistically conserved or not conserved at all.

Employing the dynamics of Eq. (36) yields a diverging kinetic energy, which results in a diverging event rate. The algorithm which we refer to as turn-at-collision-ECMC – without resetting the magnitude of velocity between chains – is valid, as demonstrated in Fig. 12. The diverging event rate renders the algorithm impractical though. The reason for this divergence is that on average each collision increases the velocity, due to the fact that there are more velocity-increasing collisions than there are velocity-decreasing

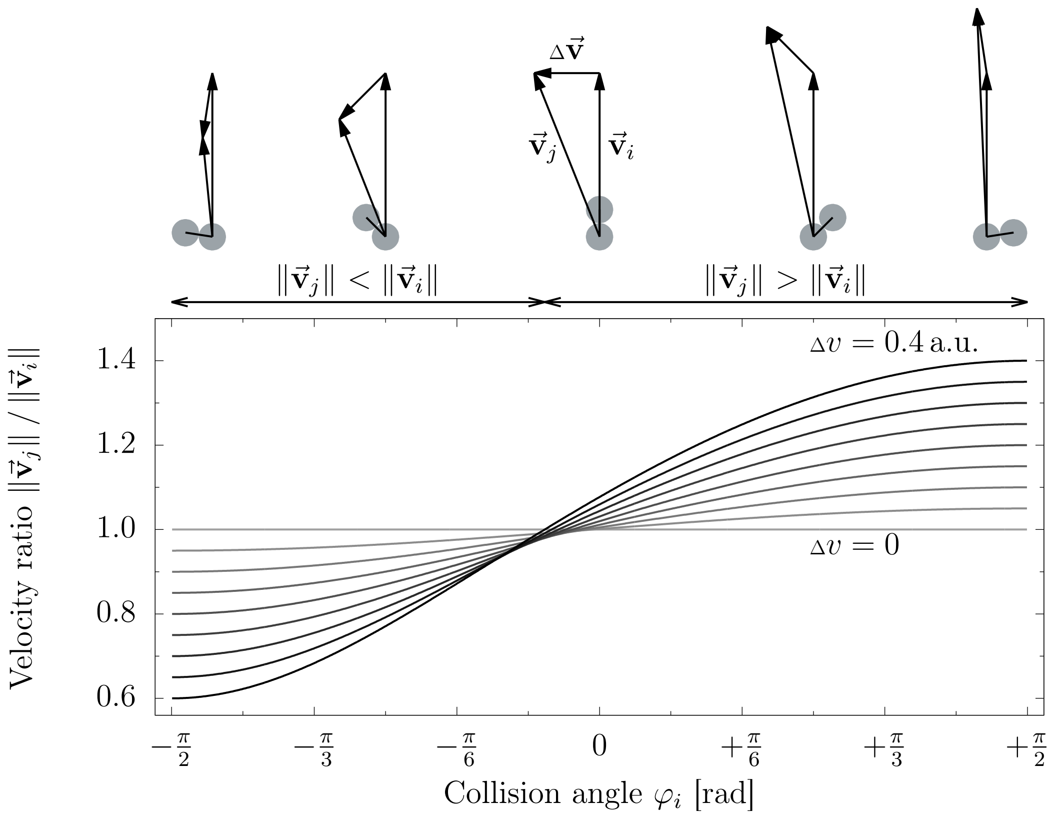


Figure 6: The velocity ratio depending on the collision angle and constant normal velocity addend as given by the square root of Eq. (32). The velocity-preserving collision angle is $-\arcsin \frac{\Delta v}{2\|\vec{\mathbf{v}}_i\|} < 0$. Hence the relative outgoing speed lies above unity for the greater portion of possible collision angles for every constant $\Delta v > 0$. The asymmetry between velocity-decreasing and velocity-increasing collisions grows with Δv. Top row pictures show incoming and outgoing velocity at different collision angles.

ones. Using the average over collision angles, see Eq. (21), and substituting $\sin \varphi_i = \zeta$, Eq. (32) yields for $\Delta v = \text{const.}$

$$\left\{ \frac{\|\vec{\mathbf{v}}_j\|^2}{\|\vec{\mathbf{v}}_i\|^2} \right\}_{\varphi_i} = \int_{-1}^{1} \left(1 + \left(\frac{\Delta v}{\|\vec{\mathbf{v}}_i\|} \right)^2 + \frac{\Delta v}{\|\vec{\mathbf{v}}_i\|} \zeta \right) \mathrm{d}\zeta = 1 + \left(\frac{\Delta v}{\|\vec{\mathbf{v}}_i\|} \right)^2 > 1 \tag{38}$$

This implies that the velocity increases on average. This circumstance is also explained with Fig. 6.

Although the velocity increase per collision diminishes with time (for diverging $\|\vec{\mathbf{v}}_i\|$ and constant Δv the right hand side of Eq. (32) converges to 1), the velocity-decreasing events never outperform the velocity-increasing ones. Furthermore, since the time budget for a single event chain stays constant, the velocity increase leads to an increasing number

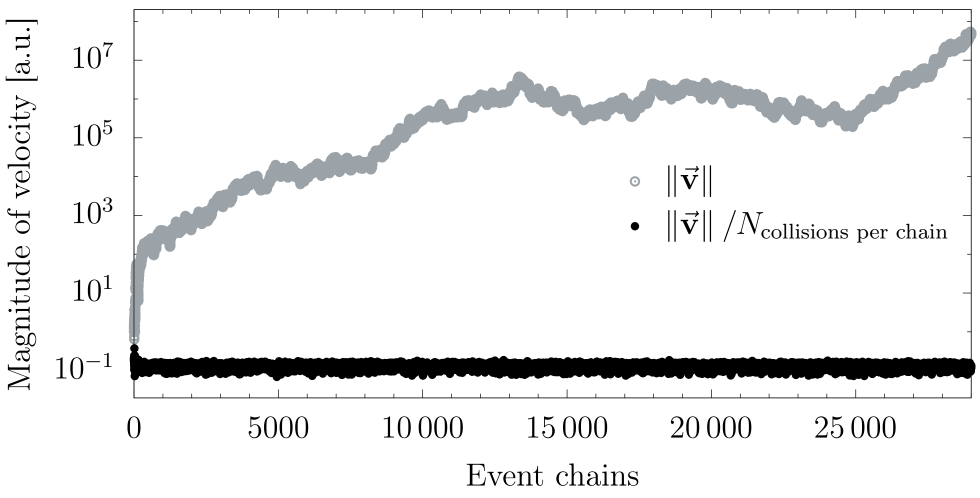


Figure 7: Evolution of velocity – mind the logarithmic scale. The current velocity divided by the number of collisions in the current chain is constant.

of events per chain; again every single event increasing the velocity on average. We find that the velocity is directly proportional to the number of lifting events per chain, see Fig. 7.

We have checked that the algorithm does not equilibrate the system, when the magnitude of velocity is rescaled to $\|\vec{\mathbf{v}}\| = 1$ after each chain, see section 5.3. This is due to the fact that $\vec{\mathbf{v}}$ is not drawn from its stationary distribution in the case of rescaling.

Another possibility is to discard the normal component of the incoming velocity, as given by Eq. (37). In this case the mapping $\vec{\mathbf{v}}_j(\vec{\mathbf{v}}_i)$ is not injective. Hence Eq. (37) violates a requirement for global balance and this algorithm is not valid. Nevertheless Eq. (37) serves as an example, because it demonstrates how the distribution of velocities can be controlled by $\vartriangle v$ and the kinetic energy is statistically conserved. In contrast to Eq. (30), where $\vartriangle v$ is added to the normal component of the whole vector $\vec{\mathbf{v}}_i$ to obtain $\vec{\mathbf{v}}_j$, now consider replacing the normal component by $\vartriangle v$:

$$\vec{\mathbf{v}}_j = \vec{\mathbf{v}}_i + \left(-\|\vec{\mathbf{v}}_i\| \sin\varphi_i + \vartriangle v_0\right) \vec{\mathbf{e}}_\perp = \vec{\mathbf{v}}_{i\|} + \vartriangle v_0 \vec{\mathbf{e}}_\perp, \tag{39}$$

where $\vec{\mathbf{v}}_{i\|}$ denotes the velocity component parallel to the bond vector $\vec{\mathbf{r}}_{ij}$ and $\vartriangle v_0$ is constant. Then the squared velocity ratio is

$$\frac{\|\vec{\mathbf{v}}_j\|^2}{\|\vec{\mathbf{v}}_i\|^2} = \cos^2\varphi_i + \left(\frac{\vartriangle v_0}{\|\vec{\mathbf{v}}_i\|}\right)^2. \tag{40}$$

Here the velocity ratio is symmetric in the collision angle and can be tuned to lie below or above unity by setting an appropriate constant $\vartriangle v_0$, see Fig. 8. In the case of large $\vartriangle v_0 \gg \|\vec{\mathbf{v}}_i\|$, one has $\{\|\vec{\mathbf{v}}_j\| / \|\vec{\mathbf{v}}_i\|\}_{\varphi_i} > 1$; thus the velocity will increase. On the other hand $\vartriangle v_0 \ll \|\vec{\mathbf{v}}_i\|$ leads to $\{\|\vec{\mathbf{v}}_j\| / \|\vec{\mathbf{v}}_i\|\}_{\varphi_i} < 1$, due to the cosine term. This way $\vartriangle v_0$ controls the average velocity.

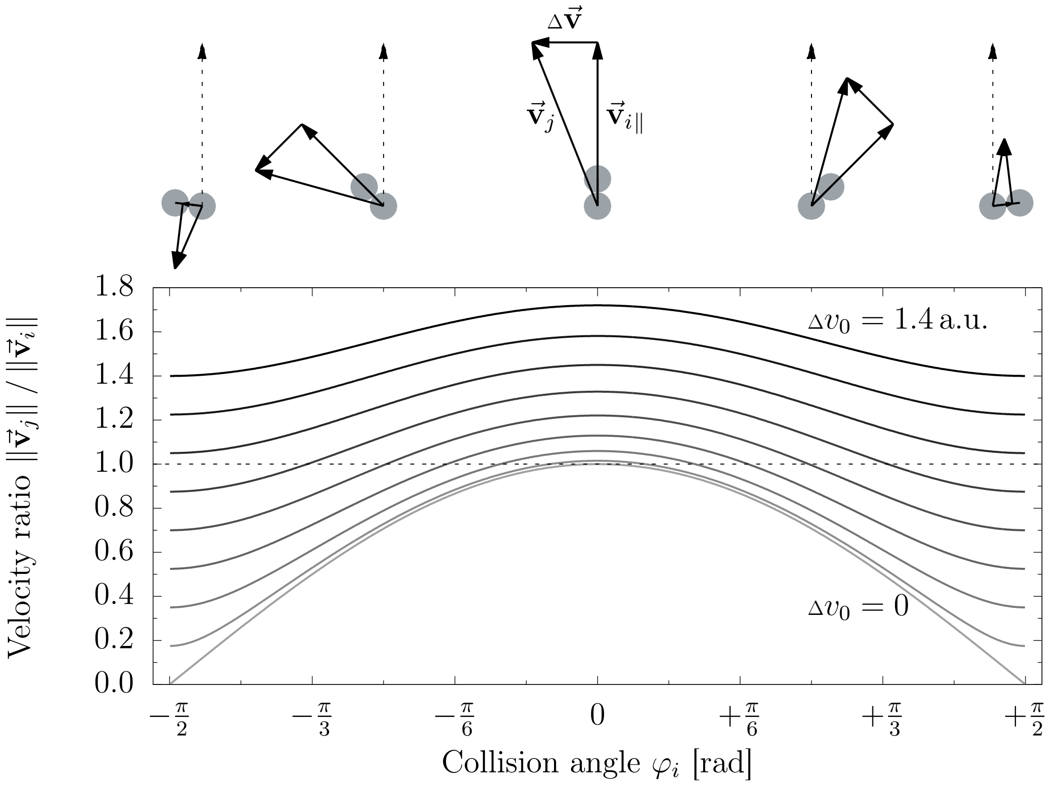


Figure 8: The $w(\varphi_i) = 1$ ratio depending on the collision angle and normal velocity addend as given by Eq. (40). Top row pictures show outgoing velocity and parallel component of incoming velocity at different collision angles; the full incoming velocity is shown dashed.

In general it should be possible to find a relation $\vartriangle v(\varphi_i, \|\vec{\mathbf{v}}_i\|)$, which yields a unity average velocity ratio, aside from the trivial solutions Eqs. (34) and (35). To this end $\vartriangle v(\varphi_i, v)$ must fulfill the balance condition

$$\left\langle \frac{\|\vec{\mathbf{v}}_j\|^2}{\|\vec{\mathbf{v}}_i\|^2} \right\rangle \overset{!}{=} 1 \quad \Leftrightarrow \quad \int_0^\infty \mathrm{d}v f(v) \int_{-1}^1 \mathrm{d}\zeta \left(v\zeta + \frac{1}{2} \vartriangle v \right) \vartriangle v \overset{!}{=} 0 \tag{41}$$

for a distribution $f(v)$. Besides the mapping $\vec{\mathbf{v}}_i \mapsto \vec{\mathbf{v}}_j$ is required to be injective. Moreover $\vartriangle v$ must not be antisymmetric in φ_i in order to set up arcuate chains. It is not obvious how to construct such a functional dependence and computing $\vartriangle v$ for every collision would certainly increase the algorithmic complexity. Furthermore an Ansatz function $\vartriangle v(\varphi_i, v)$ is not sufficient to uniquely define the distribution $f(v)$ that fulfills Eq. (41).

5.2. Arcuate ECMC turning between collisions

Since turning at collisions proves difficult, we change the moving direction independently of collisions, introducing turning events. Turning events drop out in the bulk compact initial configuration, where only instantaneous collisions of touching particles occur. In this situation the algorithms presented here are not distinct from the straight ECMC. Otherwise turning events influence the chain trajectory. A simple extension of ECMC, which we call SquareMC is discussed, followed by the more general arcuate event-chain Monte Carlo algorithm (ArcMC).

5.2.1. Square ECMC (SquareMC)

The only difference between SquareMC and straight ECMC are the dynamics of the lifting variables at the beginning of a new chain. Instead of drawing the the moving direction, *i. e.* velocity $\vec{\mathbf{v}}$, from $\{+\vec{\mathbf{e}}_1, +\vec{\mathbf{e}}_2\}$, it is drawn from $\{+\vec{\mathbf{e}}_1, +\vec{\mathbf{e}}_2, -\vec{\mathbf{e}}_1, -\vec{\mathbf{e}}_2\}$ sequentially. The first particle of the chain is chosen randomly only every fourth chain; otherwise the last particle of the previous chain is chosen. Four consecutive chains form one square chain with an approximately square-shaped trajectory, as depicted in the middle on page 3. As the excess displacement fluctuates, the sides typically do not equal exactly and the diffusive nature of trajectories in normal direction makes it improbable that the trajectory closes. When the four consecutive straight chains are considered one square chain characterized by chain displacement ℓ_{ch} and excess displacement ℓ_{ex}, it may still be appropriate to reason about the algorithm in terms of $\ell_{\mathrm{ch}}/4$ and $\ell_{\mathrm{ex}}/4$, because those quantities compare directly to the simulation box. After four straight chains another square chain starts with a random particle.

To combine several straight chains in this fashion is equivalent to introducing a *turning event*. These events occur at a certain rate (here $t_{\mathrm{turn}} = t_{\mathrm{ch}}/4$) and upon turning the moving direction is changed by an increment (here $\vartriangle\theta = \frac{\pi}{2}$).

5.2.2. Arcuate event-chain Monte Carlo (ArcMC)

We introduce a new type of event, the *turning event*, at which the direction of movement is altered by an angle increment $\vartriangle\theta$. Turning events occur at the turning event rate, of which the inverse is the turning duration t_{turn}. With a constant angle increment of fixed sign the resulting sequence of straight-line movements approximates a circular trajectory. In the limit of $\vartriangle\theta \to 0$ and $t_{\text{turn}} \to 0$ simultaneously, maintaining a constant ratio, the single-particle trajectories are circular arcs. Instead of parameterizing this new algorithm with $\vartriangle\theta$ and t_{turn}, we express $\vartriangle\theta$ in terms of t_{turn} and the gyration radius R_{G}.

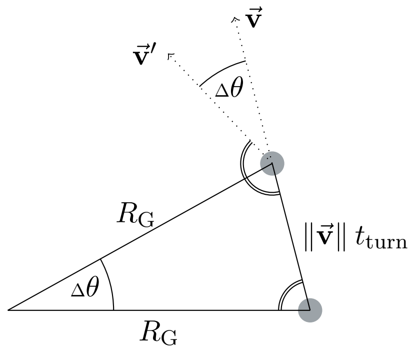


Figure 9: Effect of a turning event.

Together those parameters determine the angle increment $\vartriangle\theta$. For a single particle to move on a circle of radius R_{G} the distance $\|\vec{\mathbf{v}}\|\, t_{\text{turn}}$ travelled between two turning events must equal the length $\vartriangle\theta\, R_{\text{G}}$ of the arc of angle $\vartriangle\theta$, see Fig. 9. Yet as soon as more than one particle is involved in an event chain, the effective radius of the chain trajectory depends on the packing fraction, as every collision adds to the excess displacement. In the timespan t_{ch} the accumulated excess displacement amounts to $\langle \ell_{\text{ex}} \rangle_{\text{chains}}$ on average. Therefore the relation determining the angle increment must be

$$\vartriangle\theta\, R_{\text{G}} = \|\vec{\mathbf{v}}\|\, t_{\text{turn}} + \frac{\langle \ell_{\text{ex}} \rangle_{\text{chains}}\, t_{\text{turn}}}{t_{\text{ch}}} = \left(1 + \frac{\langle \ell_{\text{ex}} \rangle_{\text{chains}}}{\ell_{\text{ch}}}\right) \|\vec{\mathbf{v}}\|\, t_{\text{turn}} \tag{42}$$

and with Eq. (15) follows

$$\vartriangle\theta = Z \frac{\|\vec{\mathbf{v}}\|\, t_{\text{turn}}}{R_{\text{G}}}. \tag{43}$$

This angle increment is fixed throughout the simulation. Thus fluctuations of local density influence the effective radius of the chain. Already straight ECMC shows diffusive behavior orthogonal to the chain direction. Therefore the trajectory does not close, see bottom row on page 3. We define R_{G} to be the radius of chain trajectories in equilibrium. As a consequence in a void, a region in the simulation box where no particles are located, the effective radius of the trajectory is $\|\vec{\mathbf{v}}\|\, t_{\text{turn}}/\vartriangle\theta = R_{\text{G}}/Z$.

Sensible values for t_{turn}, R_{G}, and ℓ_{ch} need to be found.

We conjecture that turning events are employed most efficiently, when they occur as frequently as collision events. The number of collisions in a chain is on average

$4\ell_\text{ex}/(\pi\sigma)$. We neglect the factor $4/\pi \approx 1$ and set the number of turning events to ℓ_ex/σ. Via Eq. (15) the excess displacement can be formulated in terms of pressure. Therefore we make the choice

$$t_\text{turn} = \frac{t_\text{ch}}{\ell_\text{ex}/\sigma} = \frac{\sigma}{\|\vec{\mathbf{v}}\|\,(Z-1)}. \tag{44}$$

Indeed in equilibrium the resulting number of turning events is less than the number of collisions, but the orders of magnitude equal. In practice the equilibrium pressure Z must be known beforehand in order to compute t_turn. Is is also permissible to draw t_turn from some distribution (independent of other events, especially independent of collisions), but a constant t_turn is the simplest choice.

We assume that the optimal gyration radius R_G is about the size of a correlation domain. In the liquid phase ($\phi = 0.69$) the orientational correlation length is $\xi_6 = 11.09(20)\sigma$, as measured with the methods of [12]. In the solid phase ξ_6 diverges and a gyration radius of one or one half box length should be sufficient. Even a non-constant field $R_\text{G}(\vec{\mathbf{r}})$ depending on the location of the collision would be valid. In the limit $R_\text{G} \to \infty$ ArcMC falls back to ECMC; except for superfluous turnings by $\vartriangle\theta = 0$.

For ℓ_ch we suppose two options: First tuning $\ell_\text{ch} \sim R_\text{G}$ to achieve a certain number of revolutions of the chain trajectory. The number of revolutions is given by

$$\frac{Z\ell_\text{ch}}{2\pi R_\text{G}}. \tag{45}$$

Second, as the trajectory diffuses in radial direction, there is a prospect of coherently displacing particles in an area by tuning $\ell_\text{ch} \sim {R_\text{G}}^2$. The mean number of collisions per chain is just as for ECMC given by Eq. (22), $4(Z-1)\ell_\text{ch}/(\pi\sigma)$. We equate this to the approximate number of particles residing on average inside a circle of radius R_G

$$\phi\frac{\pi R_\text{G}^2}{\pi\sigma^2/4} = 4\phi\left(R_\text{G}/\sigma\right)^2. \tag{46}$$

The evolution of the system initialized as nearly compact crystal in a larger box demonstrates that ArcMC is capable of rotating a correlation domain. The melting of the compact crystal is structured as follows: First the surface melts, prominently at the edges of the rectangular crystal, forming a liquid halo, see Fig. 10a. The chain trajectory has a small effective radius in the sparse liquid. Therefore only few collisions but many turning events occur there. Consequently the liquid diffuses into the void very slowly. Meanwhile space is inserted into the bulk and defects are formed as can be seen in Fig. 10b. Then the bulk rotates as a whole in clockwise direction with single arcuate chains running counter-clockwise. By visual inspection we infer that this happens with a constant angular velocity, preserving the homogeneous orientation of $\psi_6(\vec{\mathbf{r}})$ across the whole bulk, as shown in Fig. 10c. Over several revolutions the bulk steadily rotates, while the liquid halo slowly diffuses outward. The diffusion of the liquid halo into the void is faster for larger R_G and smaller ℓ_ch, but qualitatively the process is equal for all aforementioned values for R_G and ℓ_ch. This demonstrates that ArcMC performs poorly at equilibrating translational degrees of freedom, but is capable of coherently rotating a correlation domain.

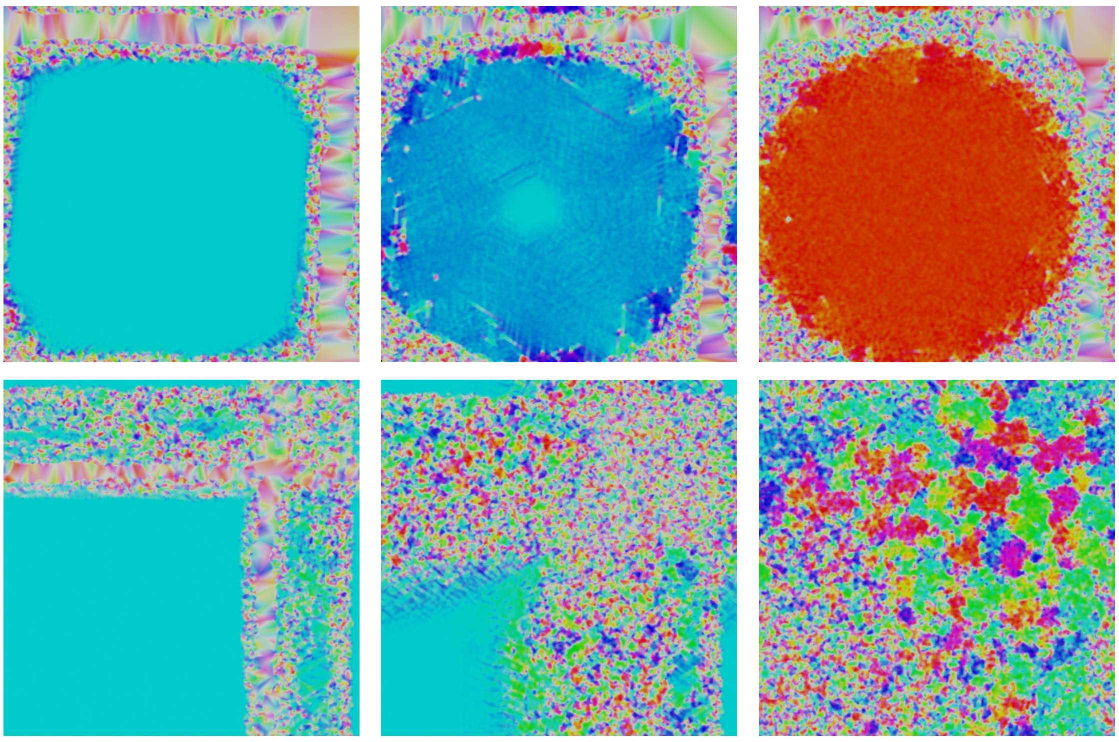

(a) (b) (c)

Figure 10: (Top row) Evolution of the nearly compact initial configuration under ArcMC. (Bottom row) For comparison the evolution under ECMC is shown, subtracted net displacement.

5.3. Validation

In order to verify that the new algorithms and their concrete implementations are correct, *i. e.* they yield the stationary distribution as the reference MMC does, we focus on the distribution of Ψ_6 in the $N = 56$, $\phi = 0.69$ system with the simulation box $\mathbb{B}$ adapted to the triangular lattice (see appendix A.1).

The following spatial symmetries imply global transformations of the bond angles θ_{ij} and corresponding transformations of Ψ_6, thus imposing symmetries on the distribution $f(\Psi_6)$:

symmetry	$\theta_{ij} \mapsto$	$\Psi_6 \mapsto$	$f(\Psi_6) =$
y-mirror	$-\theta_{ij}$	${\Psi_6}^*$	$f({\Psi_6}^*)$
x-mirror	$-\theta_{ij} + \pi$	${\Psi_6}^* \mathrm{e}^{6\mathrm{i}\pi} = {\Psi_6}^*$	$f({\Psi_6}^*)$
6-fold rotation	$\theta_{ij} + \frac{\pi}{3}$	$\Psi_6\ \mathrm{e}^{2\mathrm{i}\pi} = \Psi_6$	$f(\Psi_6)$
4-fold rotation	$\theta_{ij} + \frac{\pi}{2}$	$\Psi_6\ \mathrm{e}^{3\mathrm{i}\pi} = -\Psi_6$	$f(-\Psi_6)$
∞-fold rotation	$\theta_{ij} + \alpha$	$\Psi_6\ \mathrm{e}^{6\mathrm{i}\alpha}$	$f(\lvert\Psi_6\rvert)$

Of these symmetries the rectangular simulation box $\mathbb{B}$ possesses only the two mirrors. Hence the distribution of Ψ_6 is symmetric with respect to complex conjugation. A square-shaped box additionally possesses a 4-fold rotation symmetry, leading to an inversion symmetry in $f(\Psi_6)$. Since $\mathbb{B}$ is almost square-shaped there is an approximate inversion symmetry present. However the distribution of Ψ_6 is governed by the box shape only in small systems ($N < 10^3$). For $N = 56$ disks at packing fraction $\phi = 0.69$, the order parameter Ψ_6 has two 'attractors' on the real line, which are approximately $\pm 0.6 + 0\mathrm{i}$. The right one, at positive real part, is slightly more dominant, see Figs. 11 and 12. It corresponds to slightly distorted versions of the triangular lattice – the 7×8 arrangement fits exactly into the simulation box. However the particles may also form a 8×7 arrangement, where the lattice is rotated by $\frac{\pi}{2}$ relative to the box. As Ψ_6 is insensitive to rotations by $\frac{\pi}{3}$, its value for the 8×7 arrangement indicates a rotation by $\frac{\pi}{2} - \frac{\pi}{3} = \frac{\pi}{6}$, which corresponds to $\Psi_6 = -0.6 + 0\mathrm{i}$; see insets in Fig. 11. With larger N there are more orientations of the (deformed) lattice that fit into the box conveniently, *i. e.* are entropically favored. Thus there are more attractors, eventually forming a ring around the origin at roughly constant $|\Psi_6|$ in the hexatic phase (see Fig. 1e) or a circular cloud around $\Psi_6 = 0$ in the liquid phase respectively (see Fig. 1d).

Since Ψ_6 is an observable of the system, any valid simulation method equilibrating the system must yield the same distribution of Ψ_6. The full distribution in the complex plane looks rather noisy even for relatively long simulation runs. Therefore it stands to reason to compare the distributions of the intensity $|\Psi_6|^2$, reducing noise drastically. However a comparison of $\Re\Psi_6$ is more rigorous, because this observable is also sensitive to how long the system stays around *which* of the two attractors – a property that $|\Psi_6|^2$ lacks. Hence in Fig. 12 we use the distribution of $\Re\Psi_6$ as a simple but stringent consistency check, to prove that our implementations of ECMC and its variants do actually equilibrate the system, as the reference algorithm, MMC, does.

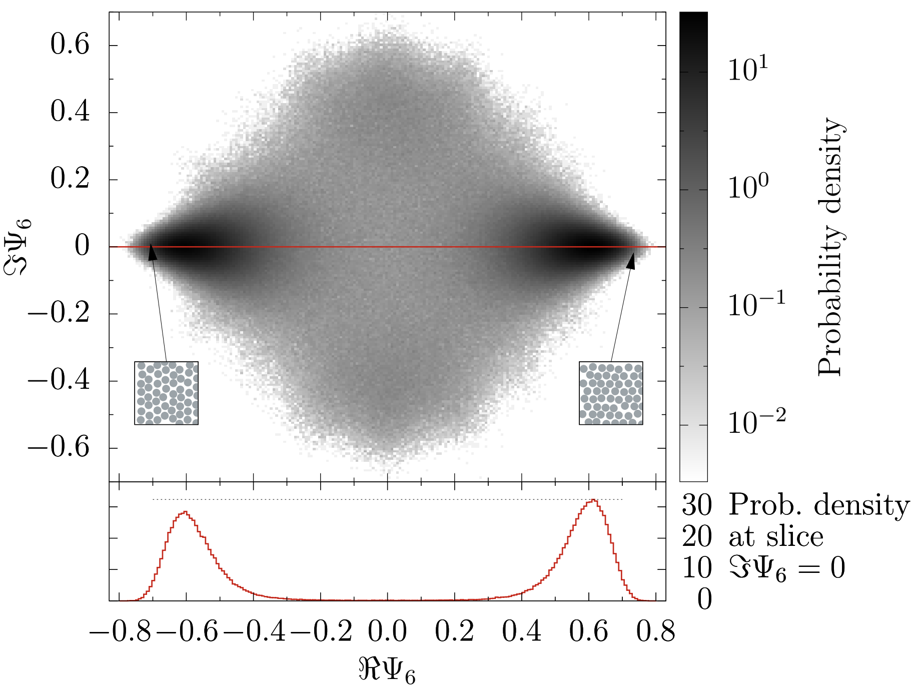


Figure 11: Probability density of Ψ_6 in a $D = 2$ system of $N = 56$ disks at packing fraction $\phi = 0.69$. Insets show exemplary disk configurations, with the arrows pointing to their respective Ψ_6 value. The twelve-digit floating-point data is binned for this grayscale histogram plot.

The cumulative distribution of $\Re\Psi_6$, sampled by an algorithm, shows three features that help discerning correct and wrong algorithms: The domain of the distribution, the slope around $\Re\Psi_6 = 0$, and the median. An algorithm not even yielding the correct domain does not explore all possible configurations, *i. e.* it is not ergodic. The slope corresponds to the probability density in the low-intensity Ψ_6 region, which is unfavored in this system, see Fig. 11. Checking the position of the median is a simple way of ascertaining that the algorithm does sample the two attractors in the correct ratio. Comparing six independent simulation runs of MMC we find that the domain is correct for all runs after 3×10^7 MMC steps, the slope is correct for all after 5×10^8 MMC steps, and median is correct for all after 5×10^9 MMC steps. The red curves in Fig. 12, representing variants of ECMC with velocity change at collision, differ from the reference curve, while the data from ECMC, square ECMC, and ArcMC collapses with the reference curve. In the case of the turn-at-collision-ECMC algorithm, the wrong median can be attributed to limited simulation time. Based on the fact that the slope is correct, the algorithm is still judged valid. In contrast the same algorithm, modified with rescalings of the velocity after each event chain, fails to yield the correct slope, even after long simulation. This proves that the modified algorithm does not equilibrate the system. An algorithm failing equilibration does not necessarily mean not converging; it may converge to a non-equilibrium steady state.

Still Fig. 12 comprises finite simulation runs, causing the distributions to deviate from each other by small amounts. Thus it remains to be shown whether these deviations can safely be considered as noise from finite sampling. Otherwise, the presence of deviations would imply that an algorithm fails to equilibrate the system. To check for this, we monitor the difference of the $\Re\Psi_6$ histograms with increasingly longer simulation runs, *i. e.* number of samples S. For a MCMC algorithm A, which is compared to the reference algorithm MMC, the difference $\Delta_A(\text{bin}; S)$ is defined as

$$\Delta_A(\text{bin}; S) := \frac{H_A(\text{bin}; S)}{H_{\text{MMC}}(\text{bin}; S \to \infty)} - 1, \tag{47}$$

with normalized histogram values $H_A(\text{bin}; S)$ including S samples obtained from algorithm A. The difference is normalized with $H_{\text{MMC}}(\text{bin}; S \to \infty)$, the reference value with best available statistics (3×10^6 samples). For any valid algorithm $\Delta_A(\text{bin}; S)$ must converge to 0 for $S \to \infty$. Moreover MCMC algorithms obey

$$\Delta_A(\text{bin}; S) \propto \frac{1}{\sqrt{S}}. \tag{48}$$

Slower decay or saturation would clearly indicate that the algorithm in question yields a different $\Re\Psi_6$ distribution, *i. e.* A would not equilibrate the system. Fig. 13 demonstrates that $\Delta_{\text{ECMC}}(\text{bin}; S)$ obeys the aforementioned power law, as it is a correct algorithm. Analogous plots for the other algorithms *viz.* square ECMC, ArcMC, and turn-at-collision-ECMC exhibit the same $1/\sqrt{S}$ behavior. Among all algorithms examined in Fig. 12 saturation is only present in the modified turn-at-collision-ECMC, confirming its invalidity.

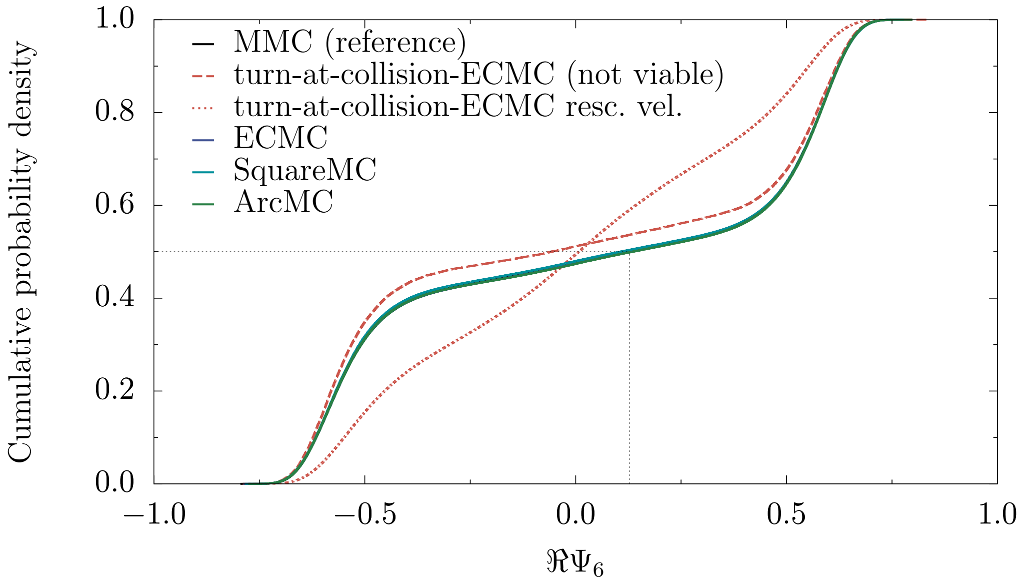


Figure 12: Cumulative probability density of $\Re\Psi_6$, obtained from different simulation algorithms. The better the statistics, *i. e.* the more data points from simulation, the smoother are the curves. In a $D = 2$ system of $N = 56$ disks at packing fraction $\phi = 0.69$. The position of the median clearly demonstrates that the attractor with positive real part is more favored. The algorithms are defined in the following sections: turn-at-collision-ECMC in section 5.1.2, Eq. (36); SquareMC in section 5.2.1; ArcMC in section 5.2.2.

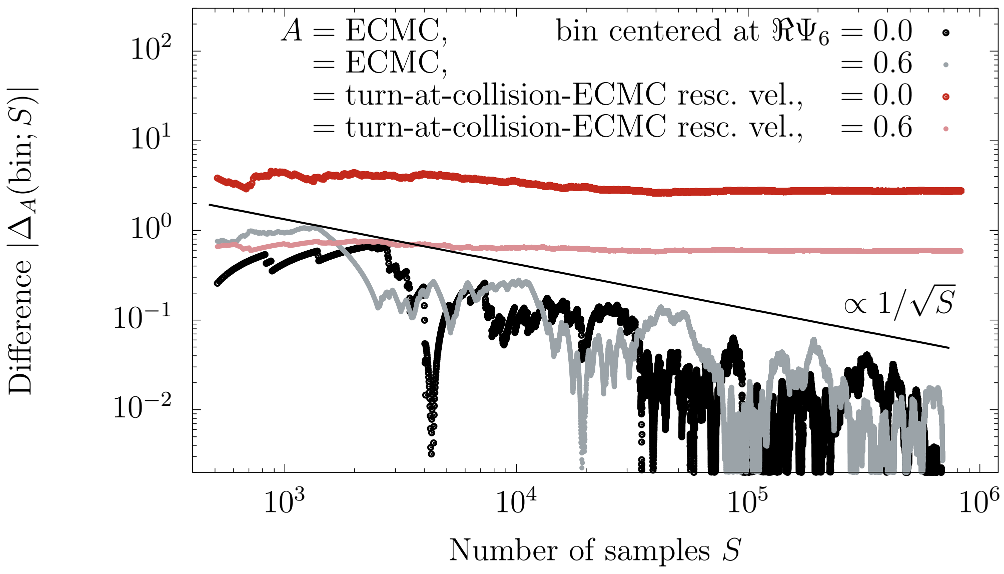


Figure 13: The histogram differences, defined in Eq. (47), between ECMC and reference (obtained from MMC) show essentially the same $1/\sqrt{S}$ decay with number of samples S at both bins. If an algorithm A fails to equilibrate, its histogram does not converge towards the reference. Then $\Delta_A(\text{bin}; S)$ does not decay, but saturates. This is the case for turn-at-collision-ECMC with velocity rescalings. These data were produced with bin width 0.05.

6. Correlation and mixing time of local Markov chains in hard disks

When comparing MCMC algorithms, the key question is, after which amount τ of simulation time a system has forgotten about its history. The wall-clock time depends of course on the hardware and software in use, but the number of events or moves gives a sufficient indication. The relevant quantity for the comparison of algorithms is the asymptotic scaling of τ with the system size N. To be able to compare the different algorithms, our notion of time is the number of "attempted one-particle displacements", as in [17]: The unit of time is for

- MMC one proposed Metropolis step (including rejected moves, as those also consume time);
- ECMC one lifting event, *i. e.* collision event or chain end event;
- SquareMC and ArcMC also each turning event counts as 1 events

This notion of discrete time is not to be confused with the time variable used to describe event chains, which connects the absolute lengths of displacements with the (extraphysical) moving velocity of ECMC. Cell-crossing events (see appendix A.2) occur roughly at rates of 0.03 moves for MMC and 0.08 lifting events for ECMC at packing fractions around $\phi = 0.7$. As those events are relatively rare and their frequencies are of the same order of magnitude for both algorithms, it is justified to ignore cell crossings in the discussion of performance. The prediction and processing of turning events is much simpler than it is for collision events. Thus it might seem overly conservative to count turning events as time steps. But on the other hand the use of turning events in ArcMC requires full vectorial treatment of $\vec{\mathbf{v}}$, prohibiting simplifications made in ECMC. On a Xeon L5520, 2.3 GHz CPU our implementation of ECMC processes roughly 7×10^{-4} collisions per clock cycle, while ArcMC has only 3×10^{-4}. Hence, as we tune the turning event rate to lie on the order of the collision event rate, it is justified to count the turning events as a pay-off for full vectorial treatment. For SquareMC there are only three turning events per chain and the same simplifications as in ECMC are made.

The scaling of τ can depend on the details of the algorithm, compare [25]. Thus we choose optimal parameter values $R_{\mathrm{MMC}} = 0.15$ and for ECMC $\ell_{\mathrm{ch}} = L_{\mathbb{B}}/Z$, as determined in section 6.3. The timescale τ can be estimated via the *empirical correlation time* τ_6 of an observable, in our case Ψ_6. Alternatively τ can be measured as the *mixing time* τ_{mix}, which characterizes the relaxation from the most unfavorable initial configuration [26].

6.1. Correlation time

The correlation time is the timescale of decorrelation of steady state equilibrium samples. Hence it is independent of the initial configuration. In $D = 2$ it is conjectured that Ψ_6 has the largest correlation time [27]. Thus, we measure τ_6 as the characteristic decay

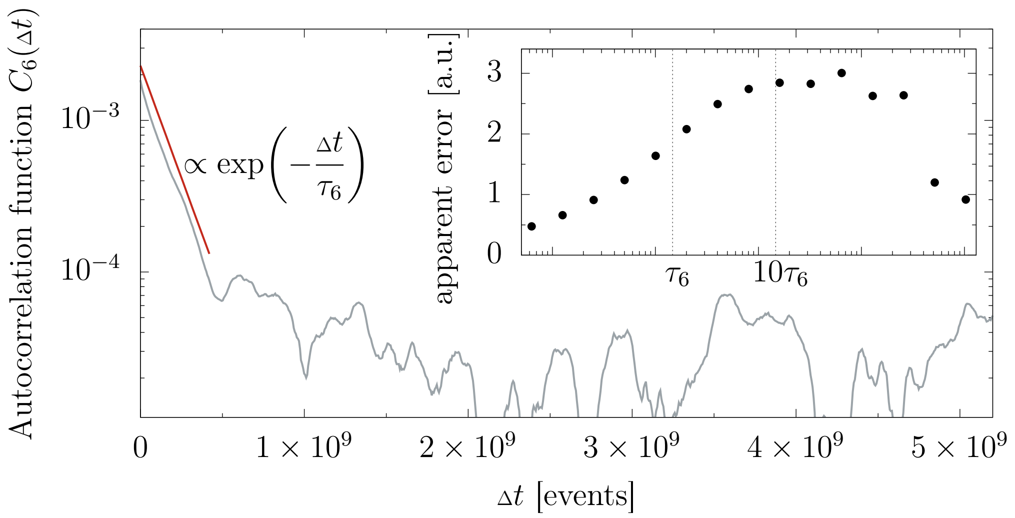


Figure 14: Autocorrelation function and exponential fit (offset in favor of visibility), by which the correlation time τ_6 is determined. The inset shows the apparent error from data bunching over the bunching interval. A plateau sets in at roughly ten times the fitted correlation time.

time of the autocorrelation function

$$C_6(\Delta t) = |\langle\, \Psi_6(t)\, {\Psi_6}^*(t + \Delta t)\, \rangle_t| \,. \tag{49}$$

Use of fast Fourier transform (exploiting the Wiener-Khinchin theorem [28]) allows for efficient computation of $C_6(\Delta t)$. The autocorrelation function starts at the value $C_6(0) = \langle|\Psi_6|^2\rangle_t$ and shows exponential decay asymptotically. The squared *empirical* mean $|\langle\Psi_6\rangle_t|^2$ is not subtracted, since the *analytical* mean vanishes in the limit of large N. The noise level of C_6 is inversely proportional to the square root of the number of samples comprising $\langle\cdot\rangle_t$. Thus a high noise level arises from taking too few samples, overshadowing the exponential decay.

The correlation time is determined by fitting an exponential function, with decay time τ_6, to $C_6(t)$, excluding the noise, see Fig. 14. As a consistency check, the fitted correlation times are compared with the rougher estimates from *bunching* [22] on Ψ_6: For all simulations, the onset of the plateau is observed at roughly ten times the fitted correlation times, see inset of Fig. 14. This fits the expectation that the plateau should set in when samples are actually decorrelated.

Fig. 15 shows the correlation times of the algorithms with optimal parameters. The corresponding power-law scaling of τ_6 and the speedup relative to MMC is given in Tab. 1. In the hexatic phase τ_6 is larger and also scales worse than in the liquid phase, namely by a factor of $\sqrt{N}$.

ECMC does outperform MMC in all cases observed. But in contrast to $D = 1$, where ECMC is faster by factor N [25], in $D = 2$ ECMC is faster only by a constant factor of 16. . . 19, not by powers of N. This is due to the fact that in $D = 2$ we measure

	$\phi = 0.69$ (lq)		$\phi = 0.71$ (hex)	
	τ_6 [events]	speedup	τ_6 [events]	speedup
MMC	$2.4 \times 10^4 N^{2/2}$	1	$3.6 \times 10^5 N^{3/2}$	1
ECMC	$1.3 \times 10^3 N^{2/2}$	19	$2.2 \times 10^4 N^{3/2}$	16
SquareMC	$3.5 \times 10^3 N^{2/2}$	7.0	$5.3 \times 10^4 N^{3/2}$	6.8
ArcMC	$3.4 \times 10^3 N^{2/2}$	7.1		

Table 1: Scaling of correlation times τ_6 with number of particles N and relative speedup $\tau_6((\text{MMC}))/\tau_6$. Power-law fits to data in Fig. 15, with exponents constrained to multiples of 1/2.

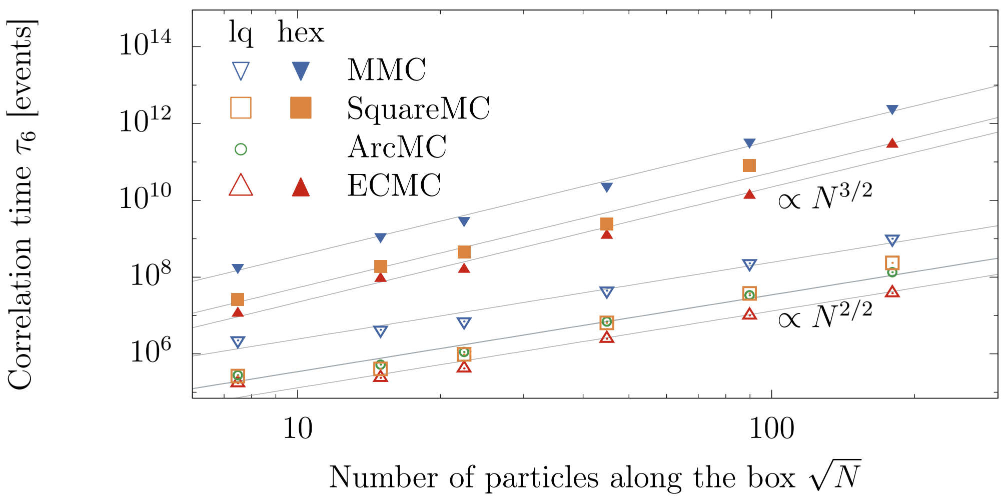


Figure 15: Scaling of correlation time with system size for different algorithms in different phases: (lq) liquid, $\phi = 0.69$ and (hex) hexatic, $\phi = 0.71$. Power-law fit values are given in Tab. 1.

the performance of the algorithms in terms of orientational order equilibration, whereas in $D = 1$ there is only positional order. The identical scaling of correlation time (up to a prefactor) for both ECMC and MMC illustrates that both algorithms are equally fast at equilibrating orientational degrees of freedom. Note that our measurement of correlation time for MMC in the high density may underestimate the actual correlation time, because more than one rotation of Ψ_6 cannot be covered by our high-density MMC simulations. In [18] a smaller factor of approximately 5 is reported in the same range of N, but for soft disks and $\phi \approx 0.698$. A factor of 28 between ECMC and MMC is reported in [13] for $N \approx 2 \times 10^5$.

The SquareMC algorithm does not reach the performance of ECMC, but is still faster than MMC. The loss of performance with respect to ECMC could be due to a cancellation effect: Parts of the square chains in SquareMC can occasionally cancel each other, because they run in opposite directions. Whenever a square chain lines up with a previous one by chance, such that the edges of the squares partially coincide, some of the one-particle displacements in the earlier chain are revoked by the current chain. Furthermore it is even possible that a square chain cancels part of itself via PBC, if the total displacement along the sides, $(\ell_{\text{ch}} + \ell_{\text{ex}})/4$, matches the simulation box side length $L_{\mathbb{B}}$.

ArcMC has essentially the same correlation time as SquareMC. As Fig. 16 shows, this is observed for $R_{\text{G}} \in \{\xi_6/2, \xi_6, L_{\mathbb{B}}/2\}$ with $\ell_{\text{ch}} \sim {R_{\text{G}}}^2$ tuned for as many collisions per chain as there are particles inside a circle of radius R_{G}, as well as for $R_{\text{G}} = L_{\mathbb{B}}/2$ with about ten revolutions per chain. From this we infer that ArcMC does not directly couple to the rotational degrees of freedom.

6.2. Mixing time

The mixing time τ_{mix} is the timescale after which equilibrium is certainly reached from any initial configuration. It is governed by the slow equilibration of exceptional initial configurations. Hence in practice, when averages are computed in long simulation runs, τ_{mix} is not as relevant as τ_6, which is smaller than τ_{mix}.

We consider two different initial configurations: The dilute *regular lattice* (Fig. 17a) and the *compact crystal* within a larger box (Fig. 17b), both based on the triangular lattice (for details see appendix A.1). In the bulk of the compact configuration every disk touches six others, so those cannot move by small finite or infinitesimal displacements. The only mobile particles are those in the surface. Hence the rate of initially mobile particles scales asymptotically with $1/\sqrt{N}$. In equilibration the surface disks must be displaced first, to make room for further rearrangements. In $D = 1$ it is clear that the most unfavorable configuration is one that leaves no gaps between the particles $1, \dots, N$ [25]. It stands to reason to speculate that the two-dimensional counterpart, the compact crystal, takes this role in $D = 2$. However we find that in the liquid phase the mixing time of ECMC is slightly shorter when the initial configuration is the compact crystal, than when it is the regular lattice. It is not the objective to find optimal simulation parameters for relaxation from the compact initial configuration, because this process is dominated by the relaxation of positional degrees of freedom.

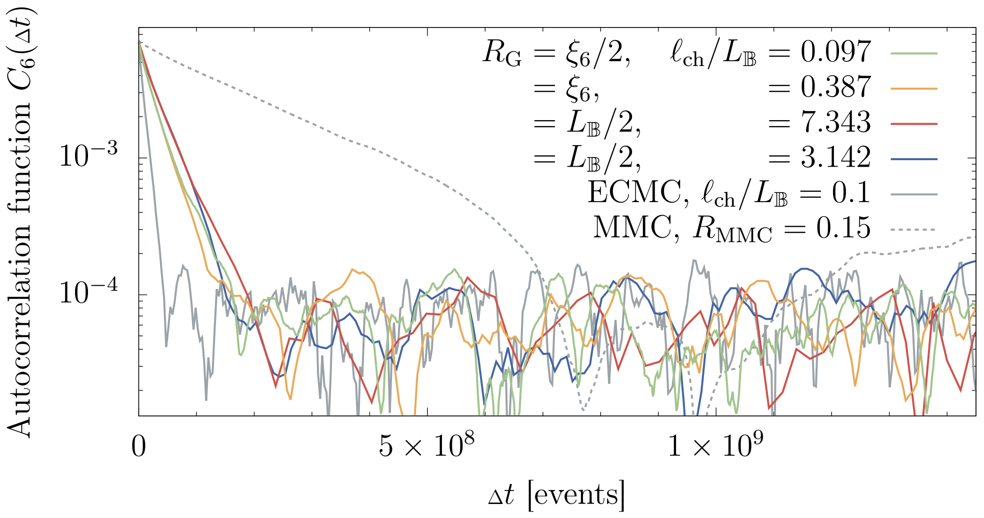
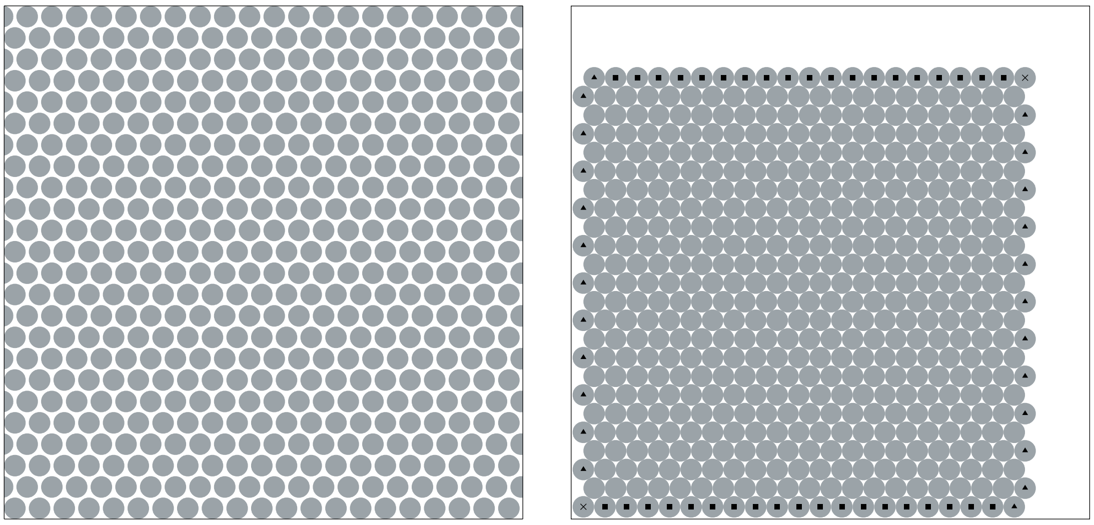


Figure 16: Autocorrelation functions for ArcMC with different parameters in $N = 8064$, $\phi = 0.69$. The correlation length is $\xi_6 \approx 11\sigma$, while the box side length is $L_{\mathbb{B}} \approx 95\sigma$. For the first three curves the chain displacement is tuned as $\ell_{ch} \sim {R_G}^2$, such that the number of collisions in a chain roughly equals the number of particles inside a circle of radius R_G, respectively. For the fourth curve $\ell_{ch} \sim R_G$ is tuned for ten revolutions per chain. The decay times hardly differ for the different parameters, while they are significantly different for other algorithms.



(a) regular

(b) compact

Figure 17: Regular and compact initial configurations for $N = 504$ and packing fraction $\phi = 0.69$. The large rectangle shows the simulation box periods in both configurations. For the compact crystal the mobile particles are marked according to their number of contacts: Those with a cross ($\times$) have two contacts, triangle (▲) three, square (■) four, and the unmarked ones have five or six contacts and cannot undergo infinitesimal displacements.

We monitor the decay of

$$C_{\mathrm{mix}}(t) := N \left|\{\Psi_6(t)\}_{\mathrm{s}}\right|^2 , \tag{50}$$

where $\{\cdot\}_{\mathrm{s}}$ denotes the average over a number of independent simulation runs; in our case at leas ten. In the liquid phase the number of correlation domains is proportional to the system size N and $|\Psi_6| \propto 1/\sqrt{N}$. This motivates the factor N in the definition of $C_{\mathrm{mix}}(t)$. In plots of $C_{\mathrm{mix}}(t)$ over t, the abscissa is rescaled with a factor N^γ to expose the timescale. Then the mixing time τ_{mix} scales as N^γ. Logarithmic corrections to the scaling, as in [25] would be visible in our data, though probably hard to quantify.

In the hexatic phase $|\Psi_6(t)|$ does not vanish with $t \to \infty$, but approaches a value that is (up to finite-size effects) independent of N. Therefore it is appropriate to consider $C_{\mathrm{mix}}(t)/N$ in this case. As soon as the system shows the final $|\Psi_6|$, the simulation runs will slowly and diffusively explore all phases $\arg \Psi_6$. Thus $C_{\mathrm{mix}}(t)/N$ does decay to zero for t tending to infinity, as it involves averaging the complex-valued Ψ_6 over independent simulation runs. However the correlation time becomes too large to simulate, such that no pronounced decay is present in our data. Instead the data from the regular lattice initialization in Fig. 18 exhibit only the scaling of a transient relaxation phenomenon. For MMC and ECMC alike this scaling is proportional to N. The mixing time cannot be smaller than the correlation time. Hence this can only be a transient scaling, as the correlation time in the hexatic phase was found to scale with $N^{3/2}$ in the previous section. The scaling with N^2 obtained from the compact initialization is marked by a dip in $C_{\mathrm{mix}}(t)$. This probably also only corresponds to a transient phenomenon: Visual inspection of snapshots reveals that the dip coincides with the filling of the void surrounding the compact crystal. The data of the hexatic phase do not allow to infer, whether the scaling is $\tau_{\mathrm{mix}} \sim N^2$ or slower.

In the liquid phase (Fig. 19), starting from the regular lattice, the results qualitatively support the conclusion of the previous section: All three algorithms MMC, ECMC, and ArcMC scale with $N^{3/2}$, differing only by a prefactor. For ArcMC we observe that the timescale obtained from the regular lattice initialization is insensitive to R_{G} and ℓ_{ch}, analogous to the correlation time. The curves $C_{\mathrm{mix}}(t)$ collapse, at least for all inspected combinations of $R_{\mathrm{G}} \in \{\xi_6/2, \xi_6, 1.5\xi_6, L_{\mathbb{B}}/2, L_{\mathbb{B}}\}$ and ℓ_{ch} tuned for a few revolutions ($\ell_{\mathrm{ch}} \sim R_{\mathrm{G}}$), as many collisions per chain as there are particles in a circle of radius R_{G} ($\ell_{\mathrm{ch}} \sim {R_{\mathrm{G}}}^2$), and $\mathcal{O}(N)$ collisions per chain ($\ell_{\mathrm{ch}} \sim N$).

With the compact configuration as starting point, ECMC does also scale as $N^{3/2}$, but reaches equilibrium even faster. This suggests that the formation of correlation domains typical for the liquid phase is enhanced, when there initially is free space, readily permitting the formation of defects. In contrast equilibration from the compact configuration reveals that the mixing time of MMC scales at least as $N^{5/2}$. We reason about this difference of a factor N in the following heuristic: Displacing a mobile particle from the compact crystal's surface across the simulation box is a diffusive process when MMC is employed. Hence the number of MMC steps needed is proportional to the mean squared displacement. The length particles must be displaced scales with $L_{\mathbb{B}} \sim \sqrt{N}$. For ECMC the displacement of particles is ballistic rather than diffusive and already the single displacements scale with ℓ_{ch}, where we choose $\ell_{\mathrm{ch}} \sim L_{\mathbb{B}}$. Therefore MMC takes

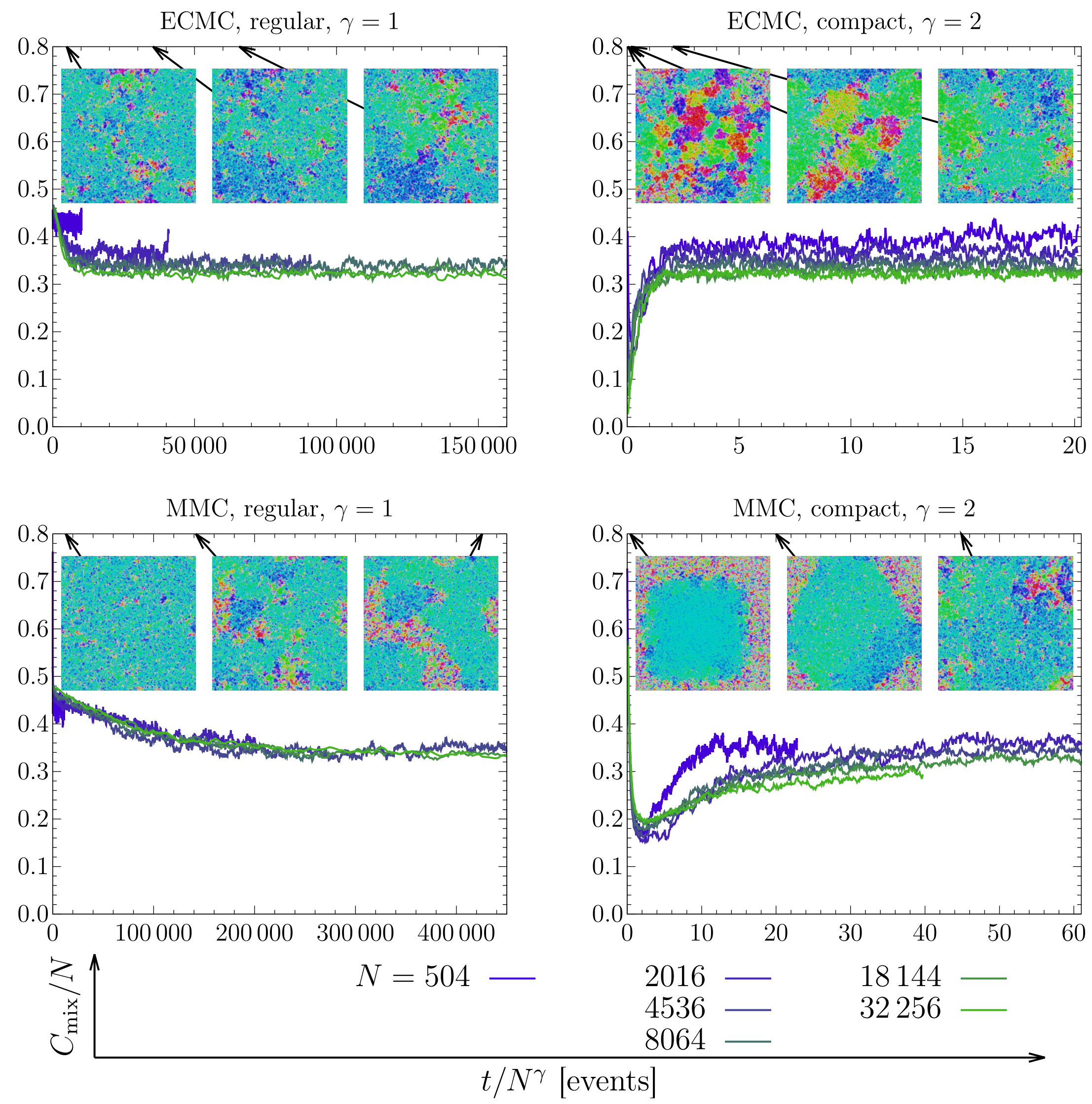


Figure 18: The scaling of $C_{\text{mix}}(t/N^{\gamma})$ in the hexatic phase ($\phi = 0.71$) from different initial configurations. Insets show snapshots of $N = 32\,256$ particles, with the arrows pointing to the respective time steps. For both initial conditions these data only expose the scaling of transient relaxation phenomena.

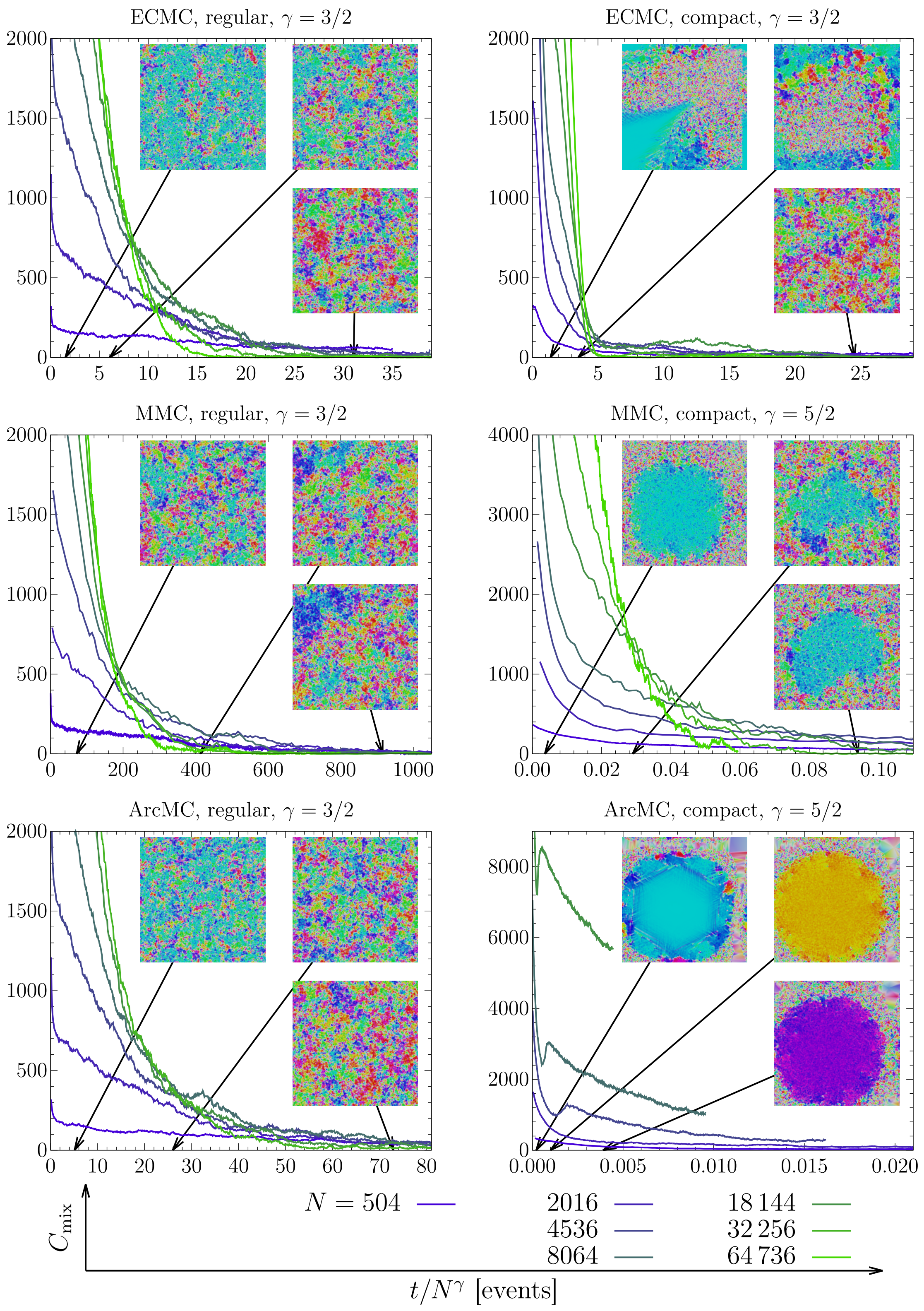


Figure 19: The scaling of $C_{\text{mix}}(t/N^{\gamma})$ in the liquid phase ($\phi = 0.69$) from different initial configurations. Insets show snapshots of $N = 32\,256$ particles, with the arrows pointing to the respective time steps.

longer by a factor of $\sqrt{N}^2 = N$. The same argument can be conducted for ArcMC. Indeed we find that ArcMC's mixing time does scale worse than that of ECMC for the range of R_G inspected. The reason for the substantially longer mixing time of ArcMC is our choice of parameters, which are not suited for quick translational equilibration. Recall that the gyration radius is the effective radius of a chain trajectory in a typical equilibrium configuration. Since the compact initialization is highly inhomogeneous, the effective radius is much larger in the dense bulk and reduced by the factor of the equilibrium pressure Z in the void. Therefore the displacement of particles into the void is not on the order of R_G, but instead on the order of the effective radius in a void R_G/Z. Consequently the filling of the void is here a diffusive process, as it is in MMC, due to the fact that we do not choose optimal parameters for equilibrating positional degrees of freedom. Accordingly we observe faster mixing for larger R_G.

Note that for ArcMC in the liquid regime, starting from the compact crystal, C_mix exhibits a dip, similar to the dip observed in the hexatic regime in Fig. 18. The position of the dip also scales with N^2 and is here associated to the formation of a liquid halo around the nearly compact bulk.

6.3. Optimal chain displacement for ECMC

For the Tonks gas ($D = 1$) it is reported in [25] that the smallest mixing time is obtained when the total displacement $\ell_\mathrm{ch} + \ell_\mathrm{ex}$ equals about half the box length. The mixing time diverges when the total displacement is an integer multiple of the box length, because in this case an event chain only results in a global translation of the configuration which cannot decrease correlations. Thus as a function of chain displacement the mixing time is oscillatory.

For hard disks ($D = 2$) we observe that the correlation time τ_6 is minimal when the chain displacement equals half of the free length $L_\mathrm{free} \approx L_\mathbb{B} - \sqrt{N}\sigma$, see Fig. 20. In isotropic systems this is roughly equivalent to equating the total displacement, $\ell_\mathrm{ch} + \ell_\mathrm{ex} = Z\ell_\mathrm{ch}$, with half of the full box length $L_\mathbb{B}$. Like in $D = 1$ a small chain displacement $\ell_\mathrm{ch} \to 0$ produces very short chains, eventually tending to single particle displacements. In this limit ECMC does not perform much better than MMC. But unlike in $D = 1$, here the correlation time shows no oscillations for larger chain displacements. τ_6 decreases monotonically for increasing $\ell_\mathrm{ch} < L_\mathrm{free}/2$ and then saturates for $\ell_\mathrm{ch} \gtrapprox L_\mathrm{free}/2$. In a two-dimensional system a straight event chain displaces only a line of particles and the chain trajectory is diffusive in the perpendicular direction. Hence a chain cannot result in a global translation of the configuration. This explains the absence of pronounced oscillations in Fig. 20.

An analysis of mixing time τ_mix for the same range of ℓ_ch confirms the above result. The data from regular lattice initialization show no dependence on the chain displacement. From the compact initial configuration, the mixing time evolves with ℓ_ch as the correlation time does. This confirms that the optimal chain displacement is $\ell_\mathrm{ch} = L_\mathrm{free}/2$. Yet for a given ℓ_ch it is easy to construct pathological initial conditions under which ECMC cannot mix. Consider for example a compact crystal initialization, with ℓ_ch exactly matching the gap between the crystal and its periodic image. In this case each one-

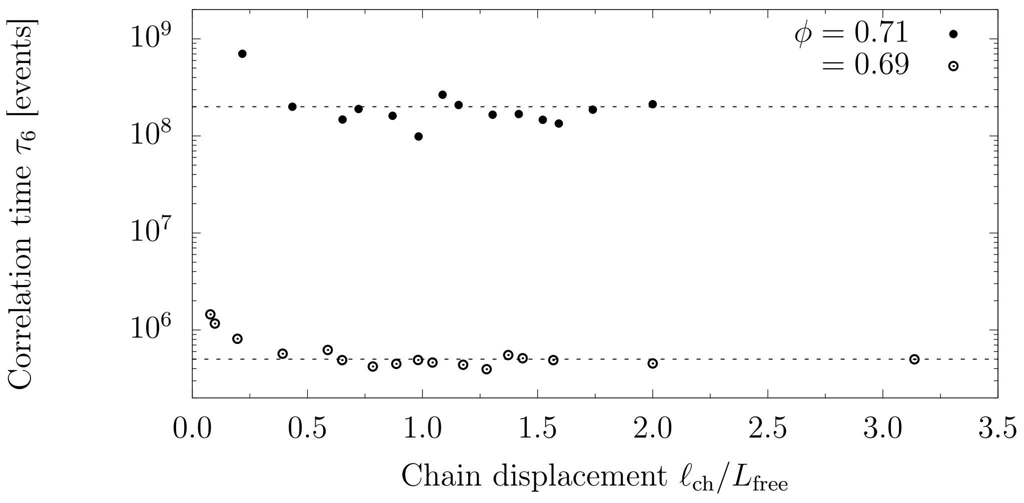


Figure 20: Correlation times for different chain displacements in the hexatic and liquid phase with $N = 504$ particles.

particle displacement takes a particle from one side of the compact crystal and attaches it to the other side – the configuration remains compact. Due to such pathological cases, the mixing time as a function of ℓ_{ch} must have poles. Though in our box geometry (see details in appendix A.1) the chain displacement cannot exactly match the free length in both directions. Hence possible divergences of mixing time are washed out.

7. Conclusion

We have substantially generalized ECMC by introducing the turning event, which permitted us to formulate ArcMC. The correctness of ArcMC and its capability of rotating a correlation domain was demonstrated. Besides, for ECMC with the velocity varying at collision, we have derived the global balance condition Eq. (28). By exploiting the freedom of choice left by the global balance condition, we have also constructed a second conceptually different arcuate ECMC algorithm, which is valid but not practically usable as the chain velocity diverges.

Furthermore the computation of pressure within the framework of ECMC was generalized for varying chain velocity Eq. (18). The distribution of collision angles was shown to be $\frac{1}{2}\cos\varphi_i$, independent of packing fraction.

The optimal chain displacement for ECMC in $D = 2$ dimensions was determined to be $\ell_{\mathrm{ch}} \geq L_{\mathrm{free}}/2$, without decrease of performance for larger values, in contrast to the established one-dimensional case. The analysis of performance revealed, that in the liquid phase the correlation time with respect to orientational order scales with the number of particles $\tau_6 \sim N$ for both MMC and ECMC. In the hexatic phase the correlation times scale as $\tau_6 \sim N^{3/2}$. In both phases ECMC outperforms MMC by a constant factor of about 16...19. The correlation time of ArcMC lies in between and is not sensitive to the tuning parameters. This suggests that ArcMC does not directly couple to the orientational degrees of freedom. All considered algorithms yielding the same scaling of correlation time implies that the algorithms are all basically equally capable of equilibrating orientational degrees of freedom. The mixing time of ECMC was found to scale at least as $\tau_{\mathrm{mix}} \sim N^{3/2}$, whereas for MMC it scales at least as $\tau_{\mathrm{mix}} \sim N^{5/2}$. We associate the slower mixing of MMC with its slower equilibration of positional degrees of freedom, which is a disadvantage when starting from a compact initial configuration. The same applies to ArcMC, where displacements across the box are unfavored due to the arcuate chain trajectories. We conjecture that a larger gyration radius improves overall equilibration from the compact configuration, but does not speed up equilibration of orientational order. Note however that the compact initial configuration is not always the worst case, which determines the mixing time. In the liquid phase ECMC can equilibrate the compact crystal faster than the regular lattice, because the initial void around the compact crystal enhances the formation of small correlation domains.

The new algorithm ArcMC may prove useful in the study of the liquid-hexatic transition of soft disks. For soft particles, *e. g.* with a r^{-6} potential, the correlation length is not only large at the phase transition, as it is for hard disks, but it even diverges. Thus an algorithm tailored for fast equilibration of orientational degrees of freedom is required. Furthermore the idea of turning events could be applied to particles with an internal rotational degree of freedom. In this case the global balance condition must be derived for the combination of translational and rotational moves, similar to the derivation of Eq. (28).

References

[1] Walter Mickel, Sebastian C. Kapfer, Gerd E. Schröder-Turk, and Klaus Mecke. Shortcomings of the bond orientational order parameters for the analysis of disordered particulate matter. *The Journal of Chemical Physics*, 138(4):044501, 2013.

[2] John M. Kosterlitz and David J. Thouless. Long range order and metastability in two dimensional solids and superfluids. (Application of dislocation theory). *Journal of Physics C: Solid State Physics*, 5(11):L124, 1972.

[3] Ze Lei and Werner Krauth. Irreversible Markov chains in spin models: Topological excitations. *EPL (Europhysics Letters)*, 121(1):10008, 2018.

[4] John M. Kosterlitz and David J. Thouless. Ordering, metastability and phase transitions in two-dimensional systems. *Journal of Physics C: Solid State Physics*, 6(7):1181, 1973.

[5] Bertrand I. Halperin and David R. Nelson. Theory of Two-Dimensional Melting. *Phys. Rev. Lett.*, 41:121–124, Jul 1978.

[6] A. Peter Young. Melting and the vector Coulomb gas in two dimensions. *Phys. Rev. B*, 19:1855–1866, Feb 1979.

[7] Bernie J. Alder and Thomas E. Wainwright. Phase Transition in Elastic Disks. *Phys. Rev.*, 127:359–361, Jul 1962.

[8] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953.

[9] Sho Asakura and Fumio Oosawa. On Interaction between Two Bodies Immersed in a Solution of Macromolecules. *The Journal of Chemical Physics*, 22(7):1255–1256, 1954.

[10] Etienne P. Bernard and Werner Krauth. Two-Step Melting in Two Dimensions: First-Order Liquid-Hexatic Transition. *Phys. Rev. Lett.*, 107:155704, Oct 2011.

[11] N. David Mermin and Herbert Wagner. Absence of Ferromagnetism or Antiferromagnetism in One- or Two-Dimensional Isotropic Heisenberg Models. *Phys. Rev. Lett.*, 17:1133–1136, Nov 1966.

[12] Sebastian C. Kapfer and Werner Krauth. Two-Dimensional Melting: From Liquid-Hexatic Coexistence to Continuous Transitions. *Phys. Rev. Lett.*, 114:035702, Jan 2015.

[13] Michael Engel, Joshua A. Anderson, Sharon C. Glotzer, Masaharu Isobe, Etienne P. Bernard, and Werner Krauth. Hard-disk equation of state: First-order liquid-hexatic transition in two dimensions with three simulation methods. *Phys. Rev. E*, 87:042134, Apr 2013.

[14] Ulli Wolff. Collective Monte Carlo Updating for Spin Systems. *Phys. Rev. Lett.*, 62:361–364, Jan 1989.

[15] Andreas Jaster. An improved Metropolis algorithm for hard core systems. *Physica A: Statistical Mechanics and its Applications*, 264(1):134 – 141, 1999.

[16] Christophe Dress and Werner Krauth. Cluster algorithm for hard spheres and related systems. *Journal of Physics A: Mathematical and General*, 28(23):L597, 1995.

[17] Etienne P. Bernard, Werner Krauth, and David B. Wilson. Event-chain Monte Carlo algorithms for hard-sphere systems. *Phys. Rev. E*, 80:056704, Nov 2009.

[18] Manon Michel, Sebastian C. Kapfer, and Werner Krauth. Generalized event-chain Monte Carlo: Constructing rejection-free global-balance algorithms from infinitesimal steps. *The Journal of Chemical Physics*, 140(5):054116, 2014.

[19] Persi Diaconis, Susan Holmes, and Radford M. Neal. Analysis of a nonreversible Markov chain sampler. *Ann. Appl. Probab.*, 10(3):726–752, 08 2000.

[20] Fang Chen, Laszlo Lovasz, and Igor Pak. *Lifting Markov chains to speed up mixing*, pages 275–281. ACM, 1999.

[21] Yuji Sakai and Koji Hukushima. Eigenvalue analysis of an irreversible random walk with skew detailed balance conditions. *Phys. Rev. E*, 93:043318, Apr 2016.

[22] Werner Krauth. *Statistical Mechanics: Algorithms and Computations.* Oxford University Press, 2006.

[23] Michael P. Allen and Dominic J. Tildesley. *Computer Simulation of Liquids.* Oxford Science Publ. Clarendon Press, 1989.

[24] Boris D. Lubachevsky. How to Simulate Billiards and Similar Systems. *Journal of Computational Physics*, 94:255–283, 1991.

[25] Sebastian C. Kapfer and Werner Krauth. Irreversible Local Markov Chains with Rapid Convergence towards Equilibrium. *Phys. Rev. Lett.*, 119:240603, Dec 2017.

[26] David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov Chains and Mixing Times.* American Mathematical Soc.

[27] Sebastian C. Kapfer and Werner Krauth. Sampling from a polytope and hard-disk Monte Carlo. *Journal of Physics: Conference Series*, 454(1):012031, 2013.

[28] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes 3rd Edition: The Art of Scientific Computing.* Cambridge University Press, New York, NY, USA, 3 edition, 2007.

[29] International Union of Crystallography. *International Tables for Crystallography*, volume A1: Symmetry Relations between Space Groups. Kluwer Academic Publishers, 2004.

[30] Hai-Chau Chang and Lih-Chung Wang. A Simple Proof of Thue's Theorem on Circle Packing. *arXiv:1009.4322*, 2010.

[31] Dennis C. Rapaport. *The Art of Molecular Dynamics Simulation.* Cambridge University Press, 2004.

[32] David Goldberg. What Every Computer Scientist Should Know About Floating-point Arithmetic. *ACM Comput. Surv.*, 23(1):5–48, March 1991.

A. Appendices

A.1. Fifty-six disks in a box

As initial configuration for monodisperse disk ($D = 2$) packings we choose the particle positions to be the lattice points of the triangular lattice (wallpaper group p6mm [29]). Only in this arrangement, the maximum packing fraction for a configuration of monodisperse disks is reached, namely $\phi_{\text{max}} = \frac{\pi}{2\sqrt{3}} \approx 0.9069$ [30]. The lattice has one particle in its primitive unit cell. It can also be constructed without any defects in an orthorhombic unit cell (the rectangular simulation box $\mathbb{B}$ under PBC) containing more particles (two at minimum), as long as the sides of the box are chosen properly. Hence for a $p \times q$ patch of the lattice, the box sides must have the aspect ratio $p : q\sqrt{3}/2$, where the factor $\sqrt{3}/2 \approx 0.8660$ is the altitude of the equilateral triangle of unity side length. We choose $p = 7$ and $q = 8$, yielding an aspect ratio of approximately $7 : 6.928$, *i. e.* an almost square-shaped box. This system of $N = pq = 56$ particles is rather small. To reach higher particle numbers, the $N = 56$ system is considered a building block for $N = 56\,n^2$ systems, $n \in \mathbb{N}$, preserving the box aspect ratio, see Fig. 21.

When a lower packing fraction $\phi < \phi_{\text{max}}$ is to be studied, we dilute the system in one of two different ways: Either a dilute regular lattice is constructed by increasing the lattice constant (or equivalently decreasing the particle diameter), as in Fig. 17a, or a compact crystal within a larger box is constructed by only increasing the box sides, as in Fig. 17b. In the latter case the symmetry of the triangular lattice is of course lost. In both cases the box sides are

$$L_{\mathbb{B},1} = \sqrt{\frac{7\pi N}{16\sqrt{3}\phi}}\,\sigma \approx 0.891\sqrt{\frac{N}{\phi}}\,\sigma \quad \text{and} \quad L_{\mathbb{B},2} = \frac{8\sqrt{3}}{14}\,L_{\mathbb{B},1}. \tag{51}$$

Their arithmetic mean is

$$L_{\mathbb{B}} = \frac{7 + 4\sqrt{3}}{8}\sqrt{\frac{\pi N}{7\sqrt{3}\phi}}\sigma \approx 0.886\sqrt{\frac{N}{\phi}}\sigma. \tag{52}$$

For the compact crystal there are 2 disks touching two neighbors, $2 \cdot 4n$ touching three, and $2 \cdot 7n - 4$ touching four neighbors, see Fig. 17b.

A.2. Cell subdivision

We implement MMC and ECMC for polydisperse hard hyperspherical particles in $D \in \mathbb{N}$ dimensions. Simulation box, particles, and cell subdivision, as described in the following, allow for arbitrary dimension D and polydispersity in particle size. Most equations given in the main text trivially generalize to the polydisperse case by substitution of $(\sigma_i + \sigma_j)/2$ for σ.

The goal of employing a cell subdivision is to reduce the algorithmic complexity of overlap tests and collision predictions. In MMC, as well as in all variants of ECMC, it is necessary to determine for one particle i if it overlaps with one of the other particles $j \neq i$, or respectively at what time particles i and j will collide. In a naive implementation

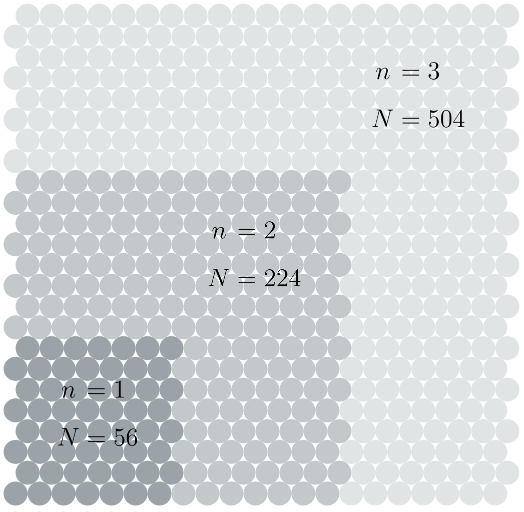


Figure 21: Colors indicate which n the disks belong to. For a given n also the (darker) disks of lower numbers belong to the respective configuration.

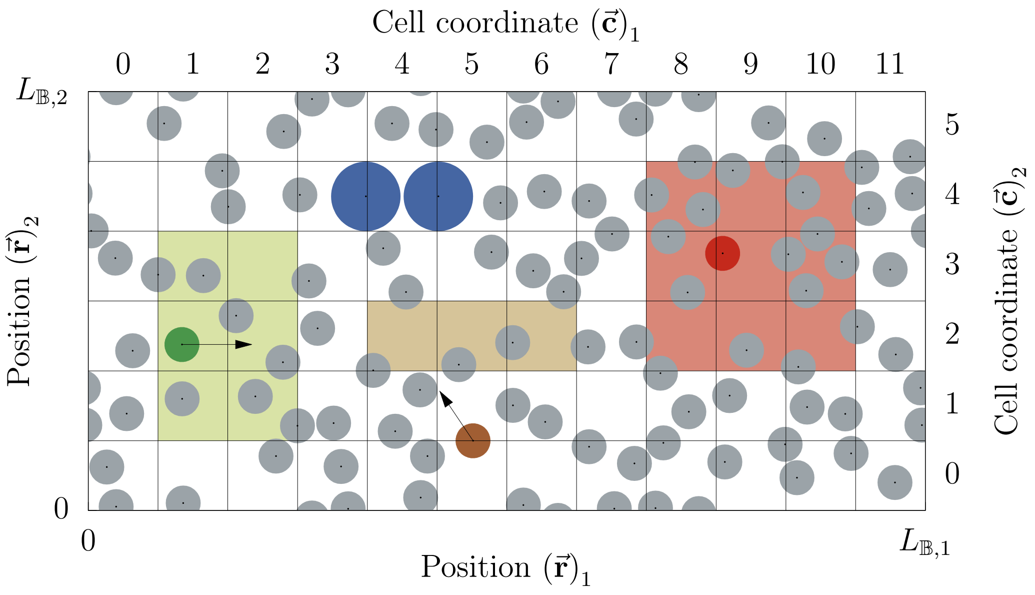
those straightforward calculations (appendix A.3) would be performed for all partners $j \neq i$, to ascertain than i overlaps with none of the others, or to find the earliest of all collisions respectively. Hence the computation of overlap or collision time for one particle is of order $\mathcal{O}(N)$. Yet particles separated by more than one diameter cannot overlap and will not collide soon. Therefore the number of viable candidates for overlaps or collisions does not scale with N, but is a constant. Hence the complexity can be reduced to $\mathcal{O}(1)$. The viable candidates are the close neighbors of particle i.

The cell subdivision is a tessellation of the simulation box into cells. It is an extraphysical data structure which only serves the purpose of keeping track of particle neighborships and handling PBC. Close neighbors of a particle are located in neighboring cells. Following [31], chapter 14, the simulation box is divided into hypercuboidal cells of equal size, shape, and orientation; $n_{\text{cells},d}$ along each dimension d; in total $\prod_{d=1}^{D} n_{\text{cells},d}$, see Fig. 22. The cells are identified[2] by their cell coordinate $\vec{\mathbf{c}} \in \{0, \dots, n_{\text{cells},1} - 1\} \times \{0, \dots, n_{\text{cells},2} - 1\} \times \cdots \times \{0, \dots, n_{\text{cells},D} - 1\} \subset \mathbb{Z}^D$. The lower boundary of cell $\vec{\mathbf{c}}$ in dimension d lies at $L_{\mathbb{B},d} \left(\vec{\mathbf{c}}\right)_d / n_{\text{cells},d}$ in position space, *i. e.* in the simulation box $\mathbb{B}$; the upper boundary lies at $L_{\mathbb{B},d}((\vec{\mathbf{c}})_d + 1)/n_{\text{cells},d}$. Particle i at

[2]The cells can be enumerated, since there is only a finite number of cells. Any cell coordinate $\vec{\mathbf{c}}$ is mapped onto a unique integer identifier $c \in \left[0, \prod_{d=1}^{D} n_{\text{cells},d}\right)$ via

$$c = (\vec{\mathbf{c}})_1 + (\vec{\mathbf{c}})_2\, n_{\text{cells},1} + (\vec{\mathbf{c}})_3\, n_{\text{cells},1} n_{\text{cells},2} + \cdots + (\vec{\mathbf{c}})_D \prod_{d=1}^{D-1} n_{\text{cells},d} \tag{53}$$

This is particularly helpful for easily storing and accessing cell-subdivision data in a fashion suited for arbitrary dimension: Instead of storing particle identifier lists (corresponding to the content of one single cell each) in a D-dimensional nested array, identifier lists are stored in one array indexed by c.



Cell coordinate $(\vec{\mathbf{c}})_1$
0
1
2
3
4
5
6
7
8
9
10
11
$L_{\mathbb{B},2}$
Position $(\vec{\mathbf{r}})_2$
0
5
4
3
2
1
0
Cell coordinate $(\vec{\mathbf{c}})_2$
0
$L_{\mathbb{B},1}$
Position $(\vec{\mathbf{r}})_1$

Figure 22: Example configuration of $N = 100$ disks in $D = 2$ dimensions. The simulation box is divided into 12×6 cells. (**top**/**blue**) The cells are chosen as large as possible, such that still the largest disks cannot touch each other unless they are located in adjacent cells. (**right**/**red**) To search for a possible overlap of the red disk with any of the other disks, only those in the highlighted 9 neighboring cells need to be examined. (**left**/**green**) When the green particle moves as indicated by the arrow, possible collision partners can only be located in the highlighted 6 cells. (**bottom**/**brown**) After a cell crossing has occurred (during the upward movement of the brown disk), when the old collision prediction was kept and updated, only particles in the highlighted 3 new neighbor cells need to be inspected for yet unpredicted collisions.

position $\vec{\mathbf{r}}_i$ resides in cell $\vec{\mathbf{c}}_i$ with

$$(\vec{\mathbf{c}}_i)_d = \lfloor n_{\text{cells},d} \, (\vec{\mathbf{r}}_i)_d \, / L_{\mathbb{B},d} \rfloor, \quad \text{for } d = 1, \ldots, D. \tag{54}$$

For gaining maximum efficiency from the cell subdivision, it is desirable to keep the list of candidates as short as possible. On the other hand the number of empty cells must be limited, since empty cells lead to useless memory accesses. Consequently a unit mean occupancy of the cells is preferred, roughly corresponding to a cell size on the order of the particle size. Nevertheless it must be impossible to forget a particle that could overlap or collide; otherwise forbidden configurations would be formed accidentally. Hence we choose the cells to be small, but still in each dimension longer than the largest diameter,

$$\frac{L_{\mathbb{B},d}}{n_{\text{cells},d}} \geq \tilde{\sigma}_{\max} \quad \Rightarrow \quad n_{\text{cells},d} \overset{!}{=} \left\lfloor \frac{L_{\mathbb{B},d}}{\tilde{\sigma}_{\max}} \right\rfloor, \quad \text{for } d = 1, \ldots, D. \tag{55}$$

More precisely $\tilde{\sigma}_{\max}$ is the arithmetic mean of diameters of the largest and second-largest particle in the system, see Fig. 22, top. This choice ensures that particle i can possibly overlap with particle j if and only if j resides in one of the 3^D neighboring cells (including the cell of i, see Fig. 22, right). Thus only the few particles in the 3^D neighboring cells need to be checked for overlap or collision. There are other implementations where cells are smaller, such that particles in *next*-nearest cells cannot touch; there 5^D cells need to be inspected and the data structure is tailored for cells only rarely containing more than one particle.

In the course of a simulation the particles leave their original cells. In order to keep track of neighborships, the cell coordinates of the particles must be updated accordingly. In MMC the cell coordinate is recomputed via Eq. (54) after every move; in event-based simulations an extra type of event, the *cell crossing*, is introduced. After a particle has crossed the boundary of its cell, the old collision prediction is kept (updated), but a search for even earlier collisions is performed. Collision partner candidates reside in the 3^{D-1} newly adjacent cells, see Fig. 22, bottom.

Similar to the updating of the collision prediction after a cell-crossing event, there are more cases in which it is unnecessary to inspect all 3^D neighboring cells. In the following two cases this is made further use of.

To check whether a configuration of N particles is entirely free of overlaps would require $\mathcal{O}\big(N^2\big)$ steps without the use of a cell subdivision. With the cells it is $\mathcal{O}(N)$. Moreover under PBC, for a complete overlap check, it suffices to inspect 2^D cells for each particle instead of 3^D. The reason is, that due to the symmetry of the overlap test only a particle's own cell and the cells in positive directions (under PBC) need to be checked.

In ECMC the velocity, *i. e.* the direction of displacements, can be chosen to be aligned to the orientation of the cells. Then there is no need to examine the particles in the 3^{D-1} cells behind the moving particle – the number of cells to search for collision partners within is $3^D - 3^{D-1} = 2 \cdot 3^{D-1}$, see Fig. 22, left. While this exclusion is valid for hard hyperspheres, it does not work out for soft hyperspheres [18]. Still even for hard regular polygons in $D = 2$ all 9 neighboring cells must be inspected, because a forward moving polygon might come into touch with another one whose center lies behind the moving particle. For algorithms with general moving directions, not aligned to the cell orientation, we speculate that it would be numerically too costly to identify which neighboring cells can be excluded from the search. Hence in those algorithms all 3^D neighboring cells are inspected, including a few cells with no relevant collision partners. Note however that after a cell boundary has been crossed in an arbitrary direction, there are always only 3^{D-1} newly adjacent cells which need to be inspected.

As used in [31], chapter 14, from a set of D integer start values s_d and end values $e_d > s_d$, a set of cell coordinate displacement vectors is spanned, namely

$$\{s_1, \dots, e_1\} \times \{s_2, \dots, e_2\} \times \cdots \times \{s_D, \dots, e_D\} \subset \mathbb{Z}^D. \tag{56}$$

The start and end values control which groups of neighboring cells are inspected. For example with $s_d = -1, e_d = +1$ the set of 3^D displacements to the neighboring cells is spanned:

$$\left\{ \begin{pmatrix} -1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ 0 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ +1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ +1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ -1 \end{pmatrix}, \dots, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ +1 \end{pmatrix} \right\}. \tag{57}$$

The 2^D displacements for an overlap test on the whole configuration are spanned by $s_d = 0, e_d = +1$:

$$\left\{ \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} +1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ +1 \\ \vdots \\ 0 \end{pmatrix}, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ 0 \end{pmatrix}, \dots, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ +1 \end{pmatrix} \right\}. \tag{58}$$

Finally for the search for collision partners in (positive) $\vec{\mathbf{e}}_1$-direction we set $s_1 = 0$, $s_d = -1$ for $d \neq 1$, and $e_d = +1$ for $d = 1, \dots, D$, resulting in $2 \cdot 3^{D-1}$ displacements

$$\left\{ \begin{pmatrix} 0 \\ -1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ 0 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} 0 \\ +1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ -1 \end{pmatrix}, \dots, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ +1 \end{pmatrix} \right\}; \tag{59}$$

and for the new collision partners after a cell crossing (still in positive $\vec{\mathbf{e}}_1$-direction) we set $s_1 = +1$, $s_d = -1$ for $d \neq 1$, and $e_d = +1$ for $d = 1, \dots, D$, which yields the 3^{D-1} vectors

$$\left\{ \begin{pmatrix} +1 \\ -1 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ 0 \\ \vdots \\ -1 \end{pmatrix}, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ -1 \end{pmatrix}, \dots, \begin{pmatrix} +1 \\ +1 \\ \vdots \\ +1 \end{pmatrix} \right\}. \tag{60}$$

Whenever a cell coordinate $\vec{\mathbf{c}}$ exceeds the range $\{0, \dots, n_{\text{cells},d} - 1\}$ in any direction $d = 1, \dots, D$ it is wrapped around subject to PBC.

A.3. Computations for ECMC and its variants

The presented event-based algorithms employ standard methods of event-driven molecular dynamics (MD) simulations. All particles follow linear trajectories and changes of velocity during a chain, if any, occur upon collisions (as in section 5.1) or turning events (as in section 5.2). The main difference to MD is that there is only one moving particle at a time, labelled by the lifting variable; all other particles rest. This circumstance simplifies the event management to great extent, as only a total of up to four events must be handled:

- cell-crossing event,
- turning event,
- collision event,
- chain-end event.

Each event is characterized by the time when the event will take place. Additionally, for a collision the collision partner must be known and for a cell crossing the dimension in which the crossing happens.

The cell-crossing event can be regarded as a bare means of bookkeeping, which is only needed to process PBC and particle neighborships. The other three types of events are lifting events, affecting the lifted configuration space, while the linear motion in between those events is a change in physical configuration space.

At the beginning of a chain the lifting variables are chosen: The time budget for the whole chain t_{ch}, the velocity $\vec{\mathbf{v}}$, and the moving particle i. Then the first events are predicted. Within a chain an a-priori unknown number of events is performed, until the pending event is the chain end; any later events are discarded. At the occurrence of an event, other events must either be updated, *i. e.* time intervals are diminished, if the old prediction is left unchanged in principle by the event, or discarded, when a completely new prediction is necessary. Consider particle i moving with velocity $\vec{\mathbf{v}}$ and the next event occurring at time interval t. Depending on the type of event, the following actions are performed:

- cell-crossing event:
  - particle i is advanced to the boundary of its new cell;
  - the cell list is adapted;
  - if necessary, PBC are applied;
  - a new cell-crossing event for particle i is predicted;
  - the turning, collision, and chain-end events are updated;
  - a search for collisions, even earlier than the updated prediction, is performed with candidate collision partners in the newly neighboring cells.
- turning event:

  - particle i is advanced to time t;
  - the velocity $\vec{\mathbf{v}}$ is altered;
  - a new turning event is scheduled;
  - the cell crossing and lifting events are discarded and repredicted based on the new velocity;
  - the chain-end event is updated.

- collision event:
  - particle i is advanced to touch its collision partner j, which will be the moving particle form then on;
  - possibly the velocity is altered;
  - the turning and chain-end events are updated;
  - new cell crossing and collision events are predicted for particle j.

- chain-end event:
  - particle i is advanced to time t.

Distinctive use is made of the terms *schedule* and *predict* an event: In the case of turning and chain-end events, scheduling an event means to draw the time of the event from a certain distribution (possibly a delta distribution). In contrast, predicting a cell crossing or collision event involves computation based on the current configuration. Moreover these computations yield, in addition to the event time, the crossing dimension or the collision partner, respectively.

For computations of cell crossing and collision events, consider particle i with diameter σ_i, currently at position $\vec{\mathbf{r}}_i$ in cell $\vec{\mathbf{c}}$, moving with velocity $\vec{\mathbf{v}}$. The time to the next cell-crossing event is the smallest t_d for $d = 1, \dots, D$ and the time to the next collision event is the smallest t_j for j labelling the candidate collision partners. The computations of t_d and t_j are described in the following sections.

A.3.1. Cell-crossing event

After a time $t_d \geq 0$, particle i crosses one of its cell's two d-faces and enters cell $\vec{\mathbf{c}} + \operatorname{sgn}((\vec{\mathbf{v}})_d)\vec{\mathbf{e}}_d$. This means that the component $(\vec{\mathbf{r}} + \vec{\mathbf{v}}t_d)_d$ has the value of the (upper or lower) d-face of cell $\vec{\mathbf{c}}$. So the crossing times are

$$t_d = \begin{cases} \left(\frac{L_{\mathbb{B},d}}{n_{\text{cells},d}} (\vec{\mathbf{c}})_d - (\vec{\mathbf{r}}_i)_d\right) / (\vec{\mathbf{v}})_d\,, & \text{if } (\vec{\mathbf{v}})_d < 0, \\ +\infty, & \text{if } (\vec{\mathbf{v}})_d = 0, \\ \left(\frac{L_{\mathbb{B},d}}{n_{\text{cells},d}} ((\vec{\mathbf{c}})_d + 1) - (\vec{\mathbf{r}}_i)_d\right) / (\vec{\mathbf{v}})_d\,, & \text{if } (\vec{\mathbf{v}})_d > 0, \end{cases} \tag{61}$$

where $\frac{L_{\mathbb{B},d}}{n_{\text{cells},d}}$ is the edge length of a single cell along dimension d.

A.3.2. Collision event

Similar to [31], chapter 14.2, for a collision with particle j (diameter σ_j, resting at position $\vec{\mathbf{r}}_j$) at a time $t_j \geq 0$, it is necessary that

$$\|(\vec{\mathbf{r}}_i + t_j\vec{\mathbf{v}}) - \vec{\mathbf{r}}_j\| \equiv \|(\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j) + t_j\vec{\mathbf{v}}\| = \frac{\sigma_i}{2} + \frac{\sigma_j}{2} =: \sigma, \tag{62}$$

where correct treatment of PBC is required in the difference $\vec{\mathbf{r}} := (\vec{\mathbf{r}}_i - \vec{\mathbf{r}}_j)$. This equation can be treated as a quadratic equation:

$$\|\vec{\mathbf{r}} + t_j\vec{\mathbf{v}}\|^2 = \sigma^2 \qquad \Leftrightarrow \qquad \|\vec{\mathbf{v}}\|^2 t_j^2 + 2\vec{\mathbf{r}}\cdot\vec{\mathbf{v}}t_j + \|\vec{\mathbf{r}}\|^2 - \sigma^2 = 0. \tag{63}$$

As only future collisions are of interest, only non-negative solutions of this equation are accepted. Purely negative solutions represent apparent collisions in the past. The existence of two solutions of different sign would imply that the two hyperspheres are overlapping initially – a forbidden configuration. If two non-negative solutions exist, the smaller one represents the actual collision event, while the larger one is the apparent collision when particle i has propagated through particle j on its "trajector[y] extended beyond the collision point" [31].

In the case $\vec{\mathbf{r}}\cdot\vec{\mathbf{v}} \geq 0$, particle i is moving away from particle j (or orthogonal to the line from center to center). Then no collision will happen. Also, if the discriminant

$$\delta := (\vec{\mathbf{r}}\cdot\vec{\mathbf{v}})^2 - \|\vec{\mathbf{v}}\|^2 \left(\|\vec{\mathbf{r}}\|^2 - \sigma^2\right) \tag{64}$$

is negative, the hyperspheres are separated too far from each other in a direction orthogonal to $\vec{\mathbf{v}}$, such that the particles will never collide. In these cases we set $t = +\infty$. Thus time t_j evaluates to

$$t_j = \begin{cases} \dfrac{\|\vec{\mathbf{r}}\|^2 - \sigma^2}{\sqrt{\delta} - \vec{\mathbf{r}}\cdot\vec{\mathbf{v}}}, & \text{if } \vec{\mathbf{r}}\cdot\vec{\mathbf{v}} < 0 \text{ and } \delta \geq 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{65}$$

The first case here is the smaller positive root, computed in a numerically stable fashion [32].

Acknowledgments

I acknowledge the continuous support of my supervisor Prof. Dr. Sebastian Kapfer. Throughout this project he was always in reach for questions regarding fine-grained details, as well as discussion on long-term plans. His motivating guidance and constructive criticism pushed the quality of this text.

I am grateful to all colleagues at theory 1 for numerous discussions – fruitful ones and the ones accompanied by fruits. In particular I thank (alphabetically and probably not comprehensively) Benedikt Decker, Sebastian Fey, Johannes Hielscher, Matthias Hoffmann, Max Hörmann, Prof. Dr. Klaus Mecke, Prof. Dr. Michael Schmiedeberg, and Felix Winterhalter for mindful listening and advice. Last but not least I am thankful to my co-tutors in the programming course: Leon Schiller, Thomas Schindler, Felix Schmidt, and Simon Weis, who bought me some hours for writing this text.

Erklärung

Ich versichere, dass ich meine Masterarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und die aus benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Erlangen, den 3. April 2018

Robert F. B. Weigel