Utrecht University

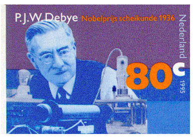
Master Thesis: Mathematical Sciences

Event-Driven Molecular Dynamics for Non-Convex Polyhedra

Author:
Marjolein van der Meer

Supervisors:
prof. dr. Rob Bisseling
prof. dr. Marjolein Dijkstra

Daily supervisors:
Simone Dussi
Nick Tasios



June 30, 2016

Acknowledgements

First of all, I would like to thank my daily supervisors Simone Dussi and Nick Tasios for their time, patience and willingness to help throughout this master project. I enjoyed our weekly meetings. I would also like to thank my supervisors Marjolein Dijkstra and Rob Bisseling for making this project possible. A special thanks goes to the master and bachelor students that shared a room with me, making it a pleasant place to work.

Abstract

Event-driven molecular dynamics (EDMD) is a technique for simulating hard particle systems. It has been effectively used to study the entropy driven self-assembly of different particle shapes, including spheres and ellipsoids. Here, we implement EDMD for hard spheres, and extend the algorithm to polyhedral-shaped particles. To confirm the correctness of our implementation, we calculate the equations of state for hard spheres and tetrahedra and compare with literature. Furthermore, the performance and accuracy of our simulation technique are studied. We proceed to study particles with a twisted triangular prism shape. This system is known to form a cholesteric liquid crystal phase. Using both twisted and non-twisted periodic boundary conditions, and by measuring the torque in the system, we directly derive the pitch of the cholesteric phase.

List of symbols

Symbol	Explanation
R	Radius
r	Position
v	Velocity
t	Time
M	Mass
N	Number of particles
E	Number of events
ω	Rotational velocity
q	Quaternion
P	Pressure
V	Volume
σ	Diameter
k_B	Boltzmann constant
η	Packing fraction
Z	Compressibility factor
T	Temperature
I	Moment of inertia
Π	Torque
$\mathcal{P}$	Cholesteric pitch
$\mathcal{P}_0$	Equilibrium pitch

Contents

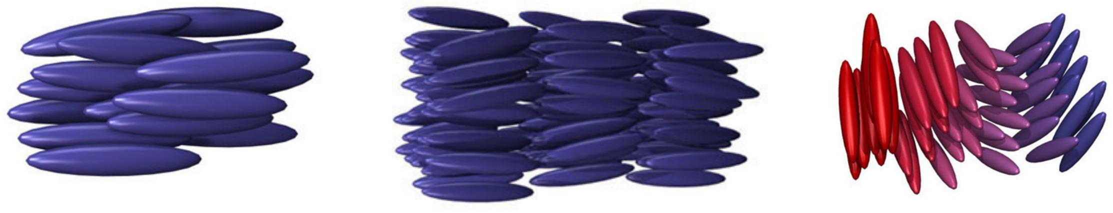
Chapter 1

Introduction

1.1 Liquid crystals

In everyday life, substances are found to be in a solid, a liquid, or a gas phase. For some substances, there is a region (of temperature and/or density) in between the solid (crystal) and the liquid phase, where they can form a liquid crystal. Here, the substance has characteristics of both a crystal and a liquid. In a crystal, molecules are ordered at regular points in space. In a liquid, molecules are moving through space, and are not ordered. In a liquid crystal, there is ordering in the orientations of the particles, but no ordering in all spatial dimensions as in a crystal phase. As a result, liquid crystals have many of the mechanical properties of a liquid, but some of the optical properties are similar to that of a crystal. This makes them suitable for many applications, the most well-known of which are liquid-crystal displays or LCDs.

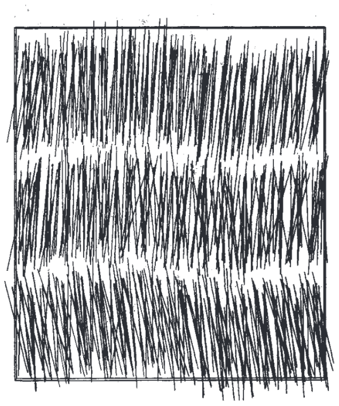
We distinguish three main types of liquid crystal phases. In the *nematic* phase, particles are only aligned along their long axis. In the *smectic* phase, in addition to the alignment of their long axes, particles are also ordered in layers. In the *cholesteric* phase, particles are aligned in such a way that a twist is produced in the system, with the particle positions homogeneously distributed. Schematics of these three types of liquid crystal phases are shown in figure 1.1, for ellipsoidal particles.



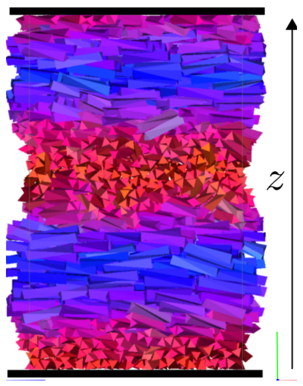
(a) Nematic phase *(b) Smectic phase* *(c) Cholesteric phase*

Figure 1.1: Schematics of different liquid crystal phases for ellipsoidal particles. In (c) particles are coloured according to their orientation to highlight the twist in the system. Source: [39].

Liquid crystals have been extensively studied in computer simulations. In particular, several studies employed hard particles (particles that cannot overlap) as simple models to study entropy-driven phase transitions. The nematic [15], smectic [13], and recently cholesteric [9] phase were all observed, two of which can be seen in figure 1.2.



(a) Smectic phase



(b) Cholesteric phase

Figure 1.2: The first smectic and cholesteric phases obtained by computer simulations of hard particles. Sources: [13, 9].

For the cholesteric liquid crystal, the particle orientations are periodic along a certain direction, as shown in figure 1.2b. The period of this twist is called the *helical pitch*.

The helical pitch of a cholesteric liquid crystal, which reflects how the molecules are microscopically oriented with respect to each other, determines properties of the liquid crystal at a macroscopic level. An interesting question is therefore how the helical pitch depends on the particle shape and the system density in the liquid crystal. *Monte Carlo* simulations have been employed to calculate the pitch [9], but another method is also possible. Given a particle shape, the pitch can be calculated exactly [16] using two different simulations of the same system of particles. The required measurements are not possible in a Monte Carlo simulation, but they are in a so-called *event-driven molecular dynamics simulation*.

1.2 Event-driven molecular dynamics

In event-driven molecular dynamics (EDMD), a number of particles is placed in a system. All particles move according to their own velocities, and interact only when collisions occur, at discrete points in time. An algorithm modelling this behaviour needs to predict collisions, and update particle positions and velocities according to the laws of physics. The termination condition is often based on a fixed number of collisions or a fixed time inside the simulation.

The first EDMD simulation was performed for a hard-disk system [1] in 1959. Since then, EDMD has often been applied to systems consisting of spheres, including spheres in a velocity field [28, 27], hard-sphere polymer chains [26], and patchy spheres [30]. The first simulation on non-spherical particles was on infinitely thin rods in 1983 [14], followed later by simulations on ellipses and ellipsoids [7], and hard cubes [21, 29].

In this report, we will apply the algorithm to so-called *twisted triangular prisms* as shown in figure 1.3. A cholesteric phase has been observed for hard particles of this shape. The goal of this thesis is to study how fast and accurately the helical pitch can be measured using EDMD simulations.

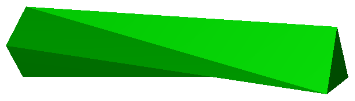


Figure 1.3: Twisted triangular prism [9].

Chapter 2

Theory & Methods

2.1 Spheres

2.1.1 EDMD

The main outline of an event-driven molecular dynamics algorithm is as follows.

1. Initialize the system.
2. Predict collisions for particle pairs.
3. Find the first collision in time, and move time forward to this point.
4. Update the positions of the colliding particles (now touching) and others, and adjust the velocities of the colliding particles.
5. Delete all predicted collisions with the particles that just collided.
6. Predict new collisions for the particles that just collided.
7. If the stop criterion is not satisfied, go to step 3.

All predicted collisions are stored. The velocities and initial positions must be such that the probability of two different collisions occurring at exactly the same time is negligible. For example, the velocities of the particles in the system can be distributed uniformly on a sphere. At the start of the algorithm, the radius R of the particles and the temperature T of the system are initialized.

The simplest version of the algorithm is the one for spherical particles, which we will start with. This is usually implemented with periodic boundary conditions, which means that if a particle leaves the simulation box at one side, it reappears at the opposite side, as can be seen in figure 2.1.

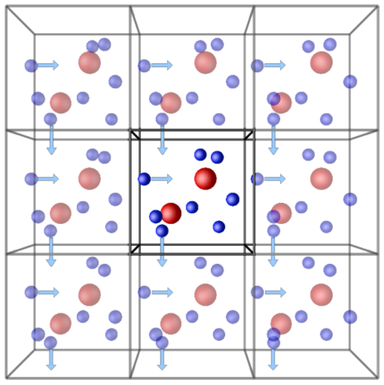


Figure 2.1: In periodic boundary conditions, the system repeats itself in all directions.
Source: [38].

2.1.2 Collision prediction

The time of collision between two spherical particles can be calculated as follows. Consider two spherical particles i and j at time t_0, with radius R_i and R_j, position $\vec{r}_i$ and $\vec{r}_j$, and velocity $\vec{v}_i$ and $\vec{v}_j$. If these two particles are to collide at time $t_0 + t$, then the following equation will be satisfied:

$$|(\vec{r}_i + t \cdot \vec{v}_i) - (\vec{r}_j + t \cdot \vec{v}_j)| = R_i + R_j. \tag{2.1}$$

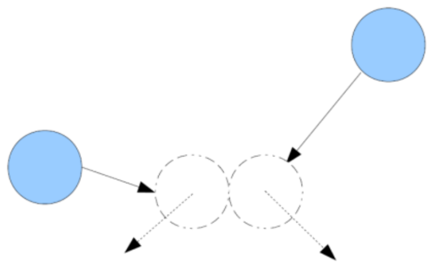
We define $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ and $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$. Then, the smallest solution to this equation is

$$t = \frac{-\langle \vec{r}_{ij}, \vec{v}_{ij} \rangle - \sqrt{(\langle \vec{r}_{ij}, \vec{v}_{ij} \rangle)^2 - \langle \vec{v}_{ij}, \vec{v}_{ij} \rangle \cdot (\langle \vec{r}_{ij}, \vec{r}_{ij} \rangle - (R_i + R_j)^2)}}{\langle \vec{v}_{ij}, \vec{v}_{ij} \rangle}. \tag{2.2}$$

In practice, we first determine the determinant under the square root. If the determinant is greater than zero, then we calculate t; otherwise, the particles will never collide (with the current velocities). Similarly, if the resulting t is not positive, the particles will never collide. A special case is when the particles are touching immediately after a collision. Due to the way velocities are updated, the two particles will not collide again.

2.1.3 Collision velocities

When two particles collide, the resulting velocities are obtained by the laws of conservation of momentum and energy. The following description is based on [36]. Consider the collision between two spherical particles i and j, with radius R_i and R_j, mass M_i and M_j, position $\vec{r}_i$ and $\vec{r}_j$ and velocity $\vec{v}_i$ and $\vec{v}_j$. Let $\vec{r} := \vec{r}_{ji}$ be the unit vector from the centre of particle i to the centre of particle j. Let $v_i := \langle \vec{v}_i, \vec{r} \rangle$ and $v_j := \langle \vec{v}_j, \vec{r} \rangle$.



By definition, v_i is the velocity of particle i in the direction of particle j before the collision, and v_j is the velocity of particle j in the same direction. Define u_i as the velocity of particle i in the direction of particle j after the collision, and u_j as the velocity of particle i in the same direction after the collision. Because the motion of each particle is purely translational, the laws of conservation of momentum and energy state that

$$\begin{cases} \text{Conservation of momentum:} & M_i v_i + M_j v_j = M_i u_i + M_j u_j, \\ \text{Conservation of kinetic energy:} & M_i v_i^2 + M_j v_j^2 = M_i u_i^2 + M_j u_j^2. \end{cases}$$

Since both particles are moving, $v_i \neq u_i$ and $u_j \neq v_j$. This leads to the following result,

$$u_i = \frac{v_i(M_i - M_j) + 2v_j M_j}{M_i + M_j}, \qquad u_j = \frac{v_j(M_j - M_i) + 2v_i M_i}{M_i + M_j}, \tag{2.3}$$

which gives the updates $\vec{v}_i = \vec{v}_i - v_i\vec{r} + u_i\vec{r}$ and $\vec{v}_j = \vec{v}_j - v_j\vec{r} + u_j\vec{r}$. Note that the velocities of the particles orthogonal to $\vec{r}$ remain unchanged.

2.1.4 Cell list

The most straightforward way of calculating the next collision in time is to calculate the moment of collision for all particle pairs (i, j) with $i < j$ and find the earliest one. However, if N is the number of particles, then this approach scales as $\mathcal{O}(N^2)$. A solution to this is to use a spatial partitioning scheme known as the cell list [27].

Consider a simulation box with periodic boundary conditions. This box is divided into a number of cells of equal size, such that at each moment in the simulation, the centre of a particle belongs to exactly one cell. For each cell, a list of the particles that belong to it is maintained. The cells are chosen in such a way that two particles can only collide if they are in the same cell, or in two neighbouring cells. This is achieved by choosing each of the sides of the cell to be no smaller than the diameter of a particle.

To check for collisions, only particles in neighbouring cells need to be considered, where periodic boundary conditions have to be taken into account as well. (This is described in more detail in [37].)

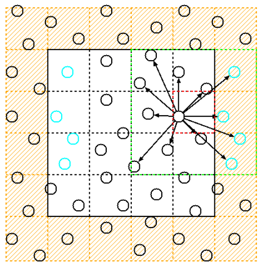


Figure 2.2: By using a cell list, less particles have to be checked for a collision. Source: [35].

Next to particle collision events, there is now a new type of event where a particle crosses from one cell to the next. This changes the algorithm in the following way. In case of a particle collision:

1. Change the velocities of the involved particles according to 2.3.
2. Remove the events associated with both particles from the stored predicted events.
3. Update the positions and local time variables of the particles in these and neighbouring cells.
4. Predict new particle collisions and the next cell boundary crossings for these particles.

In case of a cell crossing:

1. Apply periodic boundary conditions.
2. Update the particle cell coordinates.
3. Predict collisions for this particle with particles in the new neighbouring cells and the next cell boundary crossing.

By using a cell list, predicting collisions of a particle with other particles decreases from $\mathcal{O}(N^2)$ to $\mathcal{O}(1)$ with a constant depending again on the size of the cells and the density of the system.

Particles are given a local time variable, which is updated together with their position only if they are involved in or are close enough to an event. This means that for a given event, updating the particle positions is no longer $\mathcal{O}(N)$ but $\mathcal{O}(1)$ with a constant depending on the size of the cells and the density.

2.1.5 Event tree

By using an event tree, the earliest event is determined efficiently. Let E be the number of scheduled events at a given time. Using a sorting algorithm would take $\mathcal{O}(E \log E)$ time, whereas with a balanced binary tree (meaning that the number of levels is kept as small as possible) the complexity scales as $\mathcal{O}(\log E)$.

In more detail, the tree can be implemented as follows. A fixed array contains the nodes of the tree, including a pool of spare nodes. The first part of the array contains the cell crossing events, and each particle has exactly one such an event. These nodes are used for circular lists as well, such that all events associated with a given particle can easily be removed. This tree is based on [27]. However, note that in this reference, the index -1 of the array is often accessed, which would result in memory problems.

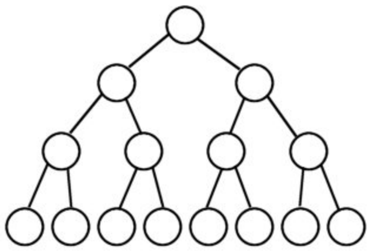


Figure 2.3: In a binary event tree, the nodes contain the events and the edges are based on the ordering in event time. Source: [34].

On top of this, the $\mathcal{O}(1)$ event tree as described in [25] can be used. This makes finding the earliest event $\mathcal{O}(1)$ instead of $\mathcal{O}(\log N)$. The idea of the algorithm is to only store events in the event tree for which the time is close enough to the current time in the simulation. If the difference between the two times is large, the event is stored in one of a few lists based on how large the difference is. If all events in the tree have been handled, the next list is emptied, the tree is filled, and this process is repeated.

2.2 Polyhedra

2.2.1 Quaternions

In order to represent the orientation of a non-spherical object in three-dimensional space, both a rotation matrix and a quaternion are a possibility. We have chosen quaternions because of the low number of calculations needed to combine two rotations for an object.

Consider a two-dimensional (infinitely thin) triangle in three-dimensional space. The position of the triangle is defined by the position of its centre of mass, as it was for spheres. The orientation can be described as follows. Define one orientation of the triangle as the base orientation. Then every other orientation can be described by a three-dimensional unit vector $\vec{v}$ around which to rotate the triangle, and a number ω that describes the angle around which the rotation should take place. Note that in this sense, an orientation is no different from a rotation.

By defining the vector $\vec{V} = \omega\vec{v}$, one only needs three numbers to describe the orientation of a particle, given the base orientation. A quaternion is a vector consisting of four numbers and hence uses an extra number to simplify multiplications. Given the angle ω and the unit vector $\vec{v}$, the corresponding quaternion is defined as

$$q_o = [s_o, p_o] = \left[\cos\frac{\omega}{2}, \left(\sin\frac{\omega}{2}\right)\vec{v}\right].$$

Note that every quaternion has norm one, which means that the space of all possible quaternions has dimension three, as expected.

Now, let the triangle be rotating. Then there is an axis represented by a vector $\vec{w}$ around which the triangle rotates, and an angle ϕ that describes the angle around which the rotation takes place in one unit of time. Hence, for a given time difference Δt, the quaternion describing the rotation is given by

$$q_r = [s_r, p_r] = \left[\cos\frac{\phi\Delta t}{2}, \left(\sin\frac{\phi\Delta t}{2}\right)\vec{w}\right].$$

Two quaternions can be combined using the following formula from [8],

$$q_f = [s_f, p_f] = q_1 q_2 = [s_1 s_2 - p_1 \cdot p_2, s_1 p_1 + s_2 p_2 - p_1 \times p_1].$$

The *rotation matrix* Q_f corresponding to q_f is defined as follows. For $p_f = (p_1, p_2, p_3)$, define

$$P_f = \begin{bmatrix} 0 & -p_3 & p_2 \\ p_3 & 0 & -p_1 \\ -p_2 & p_1 & 0 \end{bmatrix}.$$

Then

$$Q_f = 2\left[p_f p_f^T + s_f P_f + \left(s_f^2 - \frac{1}{2}\right) I\right].$$

(The difference with the formula in article [8] is intentional.) From the rotation matrix, the final orientation of a point on the particle can be obtained.

Often, a more efficient way is to use the quaternion directly. Given a three-dimensional vector $\vec{v}$ and a quaternion $q = [s, p]$, the vector $\vec{v}$ after applying the rotation described by the quaternion q is

$$\vec{v}_{\text{new}} = \vec{v} + 2p \times (p \times \vec{v} + s\vec{v}).$$

In our application, the quaternion of a particle is not always updated when its centre of mass is updated, so each particle now has two local times: one for its centre of mass corresponding to translational velocity, and one for its orientation corresponding to rotational velocity. For a random starting configuration, uniform quaternions can be used (as described in [19]), but this is not mandatory.

2.2.2 Collision prediction

Bounding spheres

For spheres, the moment of collision between two particles can be calculated exactly. For triangles, this is no longer the case. However, finding the moment of collision between two particles still corresponds to finding the first root of the function describing the distance between these particles over time. There are many root finding algorithms for problems of this form. We choose to use a method called Conservative Advancement because of its robustness. Although paper [5] shows that this method has disadvantages concerning efficiency, a root finding algorithm can miss a collision, as explained in [6].

In our simulation, the centre of mass of a triangle is defined as the centre of a sphere which exactly contains this triangle. The idea is that two triangles cannot intersect if their containing spheres are not intersecting. For a triangle pair A, B at a given time t_0, there are a few possible scenarios.

- The spheres are not intersecting. In this case, we can calculate the moment at which the two triangles will collide. There are two cases.
  - The spheres will never collide. This means that the triangles will never collide.
  - The spheres will collide at time t_1 and, without a change in velocity, leave each other at time t_2. This means that a triangle collision is only possible in the time interval $[t_1, t_2]$.
- The spheres are intersecting. In this case, we can calculate the moment t_2 at which the two spheres will leave each other. A triangle collision is now only possible in the time interval $[t_0, t_2]$.

The spheres always give us an interval in which to search for a triangle collision. The precise moment is found using Conservative Advancement.

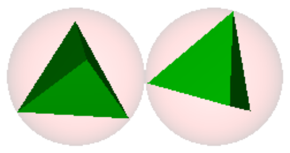


Figure 2.4: Bounding spheres of two tetrahedra.

Conservative Advancement

Conservative Advancement is a method to numerically determine the moment of collision between two moving objects, where both translational and rotational velocity are taken into account. The input consists of two convex particles with their starting positions, translational velocities, orientations, and rotational velocities. Also, a time interval in which a collision could occur is given, where a collision is defined as two objects being within a given distance $\epsilon > 0$ of each other. The method then works as follows.

Using a distance function, the distance d between the two objects is determined along with the closest points on each of the objects. Let $\vec{c}$ be the vector between these closest points. Because of the convexity of both objects, for all pairs of points on the two objects it holds that they can only collide after having travelled a combined distance d in the direction of $\vec{c}$. This information is used to determine a time step Δt to reach the first moment at which the two objects can collide. If there is a collision, then the algorithm is completed. If not, then the new positions are used to calculate a new time step, and the algorithm keeps repeating until either a collision is found, or the end of the given time interval is reached. For more details, see [32]. Since Conservative Advancement can be very slow when objects are close to each other, we convert to the Bisection Method in this case, since it is very robust. This also happens if the number of iteration steps becomes too large.

A technical detail is the following. In order to handle the collision between two particles correctly, a vector between the closest points is needed. This means that the distance between the two objects must still be strictly positive. Therefore, it may be possible that two objects are within the distance ϵ of each other, which is reported as a collision, but that their velocities are such that the closest points are moving apart (a so-called grazing collision). In this case, the velocities of the particles do not change during a 'collision'.

The exact distance between two triangles can be found using [10]. Conservative Advancement only works for convex objects. For non-convex objects, one can divide the objects into convex parts and compare those. We have chosen to divide the surface of the objects into triangles, but other choices are possible. A separate Conservative Advancement algorithm was made for objects consisting of triangles, based on pair-wise comparisons.

2.2.3 Collision velocities

Moment of inertia

In calculating new velocities for two colliding particles, their moments of inertia are needed. The inertia matrix can be calculated for an arbitrary polyhedron, as proven in [11] in combination with [24]. However, for simplicity we assume a spherical moment of inertia. The reason is that we are interested in the equilibrium properties of the system, not in the precise dynamics. Moreover, the second approach is faster and easier to implement.

Collision resolution

Resolving a collision involving torques is described in [23]. Let i and j be two colliding particles with mass M_i, M_j and spherical moment of inertia I_i, I_j. Let $\hat{n}$ be a unit vector normal to the point of contact in the direction of particle i, let v_n be the velocity with which the two particles collide in the direction of $\hat{n}$, and r_{Ci}, r_{Cj} the positions of the collision points on their corresponding objects. Then, in order to have conservation of momentum and kinetic energy, the exchange of momentum between the particles Δp_{ij} is described by

$$\Delta p_{ij} = 2v_n \left(\frac{1}{M_i} + \frac{1}{M_j} + \frac{\|r_{Ci} \times \hat{n}\|^2}{I_i} + \frac{\|r_{Cj} \times \hat{n}\|^2}{I_j} \right)^{-1}. \tag{2.4}$$

(The difference with the formula in article [23] is intentional.) For particles that do not have a spherical moment of inertia, equation 2.4 is more complicated. In both cases, the velocities and angular momenta of the particles are updated as follows.

$$\vec{v}_i \leftarrow \vec{v}_i - \frac{\Delta p_{ij}}{M_i}\hat{n}, \tag{2.5}$$

$$\vec{v}_j \leftarrow \vec{v}_j + \frac{\Delta p_{ij}}{M_j}\hat{n}, \tag{2.6}$$

$$\vec{w}_i \leftarrow \vec{w}_i - \frac{\Delta p_{ij}}{I_i}(r_{Ci} \times \hat{n}), \tag{2.7}$$

$$\vec{w}_j \leftarrow \vec{w}_j + \frac{\Delta p_{ij}}{I_j}(r_{Cj} \times \hat{n}). \tag{2.8}$$

For more details, see appendix A.

2.3 Boundary conditions

In all EDMD simulations, the simulation box only has a finite size. This means that a choice has to be made concerning the boundary conditions. We will describe three possibilities.

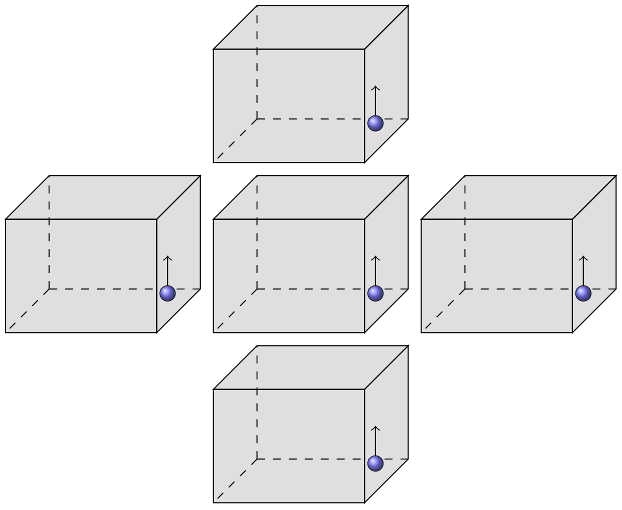


Figure 2.5: In normal periodic boundary conditions, the simulation box is surrounded by its identical periodic images.

Event-driven molecular dynamics simulations can be implemented using *periodic boundary conditions*. This means that if a particle leaves the simulation box at one side, it reappears at the opposite side with exactly the same velocity (translational and, if relevant, rotational). A characteristic of this is that a side of the system is always influenced by the side opposite to it, as shown in figure 2.5. If this is not desirable, then another possibility is to place *hard walls* in one or multiple direction(s). If a particle comes in contact with this wall, then it bounces back following similar physical laws as in a collision between two particles.

The third possibility is less intuitive. In order to determine the helical pitch of a cholesteric liquid crystal, so-called *twisted periodic boundary conditions* in one direction are needed. The result is that all sides of the system are influenced by the opposite side just like before, but in one direction, a side is influenced as if the opposite side were rotated with an angle of $\pi/2$ as illustrated by figure 2.6. The twist in the boundary conditions has implications for positions and orientations as well as translational and rotational velocities. For more information, see [2]. Twisted periodic boundary conditions are used in order to study the twist of a cholesteric liquid crystal through a torque measurement, as we will show later.

2.4 Model

In simulations, we can study situations which are very hard to produce in controlled experiments, to gain insight in the system we are studying. In this chapter we discuss the reliability of our results. How much information is lost in going from the physical system to the model? This discussion is inspired by the finite size effects shown in [4] and [12].

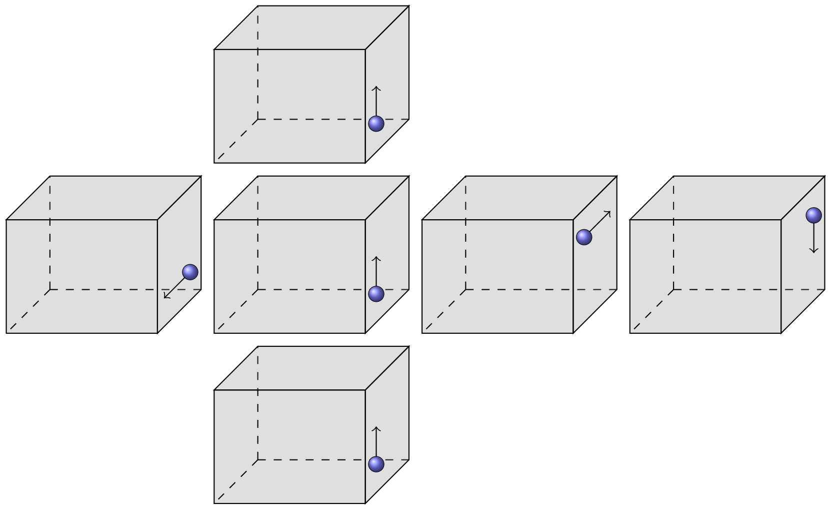


Figure 2.6: In twisted periodic boundary conditions, the simulation box is still surrounded by its periodic images, but these are rotated in one direction with respect to each other.

2.4.1 Simplifications

We begin by enumerating some of the simplifications we made in going from the physical system to the model. The first one is that in event-driven molecular dynamics, all particle interactions are assumed to be hard collisions. For non-spherical particles, a further simplification is added to this. To save computation time, the moment of inertia of all particles is assumed to be that of a sphere, independent of the particle shape. Especially for elongated particles, this changes the way collisions are resolved (for more details, see appendix A).

Another simplification that requires special attention is the periodic boundary conditions.

2.4.2 Periodic boundary conditions

As we cannot model the actual macroscopic system, periodic boundary conditions give us a way of imitating a large system while using only a limited number of particles. This way, there is no actual surface. However, this does not mean that there are no finite-size effects anymore. In the case of event-driven molecular dynamics, the finite-size effects are not as large as for some other systems because the interactions between particles are not long-ranged. However, particles transfer momentum to each other through collisions, and at some point, a particle will receive some of its own momentum back again.

Another characteristic of periodic boundary conditions is the imposed symmetry. This is not a problem for an isotropic simulation, in which case cubic boxes are the easiest way of implementation. For non-isotropic systems, there are other possibilities. Firstly, there are box shapes that give a larger distance between particles and their own periodic images, compared to the cubic box. See, for example, [31]. Secondly, walls can be used instead of periodic boundary conditions, but as mentioned before, this can result in surface effects. An example can be found in [9].

For cholesteric liquid crystals, ideally, the periodic boundary conditions should be such that the cholesteric pitch is not distorted. Unfortunately, cholesteric pitches are usually very long compared to the size of a particle. Interestingly, this is actually what we make use of in the torque measurements, so the influence of boundary conditions should not be underestimated.

2.4.3 Pressure equilibration

In the type of simulations described in this report, measurements are only reliable if they are done in equilibrium. This means that for more particles and higher densities, more collisions are required before a reliable measurement can be made. In Molecular Dynamics simulations, the rate of pressure equilibration is determined by the transfer of momentum between particles, and this is typically fast. In Monte Carlo simulations, pressure equilibrates through diffusion, which is slower. Hence, once a configuration in equilibrium has been obtained from a Monte Carlo simulation, the pressure can be measured relatively fast in an EDMD simulation.

2.4.4 Rounding errors

For all computer simulations, the possible influence of rounding errors should be considered. In this case, rounding errors and more generally numerical approximations influence the evolution of the system at hand. The many-body systems we are studying are chaotic, which means that a small difference in the simulation will grow exponentially as the simulation continues. For example, if the direction of the velocity of a particle is slightly off, then it can miss a collision, which changes not only the trajectory of this particle but also of the particle it would be colliding with, which in turn influences other particles, etcetera.

The example shows that Molecular Dynamics does not exactly predict how the actual system will evolve over time, but it is a good approximation of a possible evolution. By using statistical averages such as pressure, the results are still reliable.

2.5 Measurements

2.5.1 Equation of state

We begin with some definitions. Consider a system of N hard particles with volume v_0 in a simulation box of volume V, at pressure P and temperature T. With k_B the Boltzmann constant, the *packing fraction* η and *compressibility factor* Z are defined by the following formulas.

$$\begin{aligned} \eta &:= \frac{Nv_0}{V}, \\ Z &:= \frac{PV}{Nk_BT}. \end{aligned}$$

An EDMD simulation is in so-called *NVT ensemble*, which means that throughout the simulation, the number of particles, the volume of the simulation box and the temperature of the system are constant. Hence, the packing fraction is constant as well. (In particular, for a system of hard spheres with diameter σ, the packing fraction is $\eta = \frac{\pi}{6}\frac{N\sigma^3}{V}$.) The temperature T is fixed by

$$k_B T = \frac{2}{d}\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{2}M_i v_i^2 + \frac{1}{2}I_i\omega_i^2\right)$$

where d is the number of degrees of freedom of the particle, N is the number of particles, and M_i, v_i, I_i and ω_i are the mass, velocity, moment of inertia and rotational velocity, respectively, of particle i. (A spherical particle has 3 degrees of freedom, and a polyhedron has 6 degrees of freedom.) Since there is conservation of kinetic energy at each collision, the temperature remains the same throughout the simulation.

Contrary to the packing fraction and the temperature, the pressure is unknown at the start of the simulation. It is obtained as follows. At the start of the simulation, a variable P_m is set to zero. Then, during the time in the simulation in which the pressure is measured, each collision between two particles i and j contributes a term to this pressure. More specifically, if $\vec{r} = \vec{r}_i - \vec{r}_j$ is the vector between the centres of the spheres and $\vec{v} = \vec{v}_i - \vec{v}_j$ is the relative velocity, then a term $\vec{r}\cdot\vec{v}$ is added to P_m. At the end of the simulation, the average pressure $\bar{P}$ is

$$\bar{P} = \frac{-P_m}{3tV}$$

where t is the time in the simulation in which the pressure was measured [17].

Given the *number density* $n = N/V$, the compressibility factor Z is given by

$$Z = 1 + \frac{P}{nT}$$

where the constant term comes from the contribution of an ideal gas. By changing the number of particles and/or the volume of the simulation box, and keeping the temperature fixed, one can plot the compressibility factor against the packing fraction.

2.5.2 Cholesteric pitch

The torque in a system can be measured as follows. At the start of a simulation, let $M_{\alpha\beta} = 0$ for $\alpha, \beta = x, y, z$. For every collision, add terms to this matrix. If particle i and j collide with each other, let $r_{ij} = r_i - r_j$ be the vector between the centres of mass of the two particles. Let τ_{ij} be the torque that particle j exerts on particle i, which is defined as

$$\tau_{ij} = \Delta p_{ij}(r_{Ci} \times \hat{n}), \tag{2.9}$$

where Δp_{ij} is the exchange of momentum between the particles in equation (2.4), r_{Ci} is the position of the collision point on particle i, and $\hat{n}$ is the unit vector normal to the point of contact pointing towards particle i.

At this collision, let r_{ij} be the vector from the centre of particle i to the centre of particle j, of which r_{ij}^{α} is the entry of the α-coordinate. Add a term $r_{ij}^{\alpha}(\tau_{ij}^{\beta} - \tau_{ji}^{\beta})$ to $M_{\alpha\beta}$ for $\alpha, \beta = x, y, z$. Finally, the returned torque $\Pi_{\alpha\beta}$ is

$$\Pi_{\alpha\beta} = -\frac{1}{2}M_{\alpha\beta}$$

for $\alpha, \beta = x, y, z$.

Twisted periodic boundary conditions are used in order to study the twist of a cholesteric liquid crystal through a torque measurement, as described in [16]. Suppose that two simulations have been performed, one with normal and one with twisted periodic boundary conditions in the α-direction. Let $\Pi^n_{\alpha\alpha}$ be the measured torque in the simulation with normal periodic boundary conditions, and $\Pi^t_{\alpha\alpha}$ the same measurement for twisted periodic boundary conditions. Then the equilibrium cholesteric pitch is equal to

$$\mathcal{P}_0 = -\pi l_\alpha / 2 \cdot \frac{\Pi^n_{\alpha\alpha}}{\Pi^t_{\alpha\alpha} - \Pi^n_{\alpha\alpha}}, \tag{2.10}$$

where l_α is the length of the simulation box in the direction of α, as derived in [16].

2.6 Speed-up

2.6.1 Bounding boxes

If particles are very long and thin, then the cell list becomes inefficient. The reason is that particles can have many neighbouring particles within a radius $\epsilon > 0$ of its bounding sphere, even though the number of neighbours within a distance ϵ of the actual particle is a lot less. (Note that for spheres, this number would be exactly the same.) In order to improve on the running time of the simulation, we used *nearest neighbour lists*, loosely based on [8].

The idea is the following. At the start of the simulation, each particle is placed in an imaginary box, which completely encloses the particle and does not touch it. This box stays fixed, independently of the movement of the particle inside it. Two particles are considered neighbours if their bounding boxes overlap. During the entire algorithm, only collisions between neighbours are recalculated.

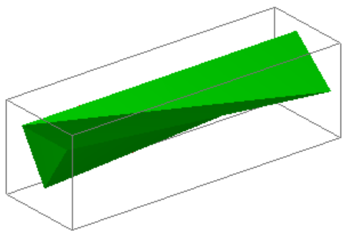


Figure 2.7: Bounding box of a twisted triangular prism.

Apart from bounding sphere collisions, particle collisions and cell crossings, there is now another type of event, which is when a particle collides with its bounding box. When this happens, a new box is built around the particle. The cell list is used to recalculate the neighbours. The more a neighbour list can be used before the particle collides with its bounding box, the higher the improvement in efficiency.

2.6.2 Parallelization

Even without their elongated form, twisted triangular prisms are computationally expensive because each particle consists of 8 triangles, and our implementation of the Conservative Advancement is not constant (but less than quadratic) in the number of triangles. In the hope of improving on the running time of the simulation for large systems, we implemented a parallel algorithm based on [22] using domain decomposition.

At the start of the simulation, each processor gets some of the cells in the system, which it will handle during the rest of the simulation (note that we do not use dynamic load balancing). For a given processor P, particles in cells on this processor are called *real* and particles in cells on other processors *virtual*. Processor P only calculates collision times for particle pairs of which at least one particle is real. If a particle moves from a cell on processor P to a cell on another processor, P communicates this particle to that processor and removes the events of the particle from its own event tree. Similarly, P can receive particles.

Contrary to the algorithm described in [22], there is communication after each event, where the processor on which the event occurred communicates the changed particles to all processors that contain neighbouring cells. This communication is expensive. To still obtain an improvement in running time, we let the processor handling an event only recalculate collisions between two real particles, and leave all other collisions for other processors to calculate.

2.7 Restarts

During a simulation, the program will output a saved state regularly. Each saved state is a file containing all the information of the simulation at this point in time. The printed characteristics are described in more detail in appendix C.

Saved states can be used to study the behaviour of the simulation over time. Given a saved state, the simulation can also be restarted from this information. This is useful if the simulation terminates prematurely, for whatever reason, in which case the simulation can be restarted from the last saved state.

Chapter 3

Results & Discussion

3.1 Validation

3.1.1 Spheres

We first wrote an EDMD simulation for spheres, based on the theory described in the previous chapter. In order to test the correctness of our simulation, we compare our results with the literature.

There are two equations relating the packing fraction and the compressibility factor of a system. The *Carnahan-Starling equation of state* is reliable for low packing fractions and defined by

$$Z = \frac{1+\eta+\eta^2-\eta^3}{(1-\eta)^3}.$$

The *Speedy equation of state* is reliable for high packing fractions and states that

$$Z = \frac{3}{1-z} - \frac{a(z-b)}{z-c}$$

where $z = (6\eta)/(\pi\sqrt{2})$ and a, b, c are constants. According to [3], $a \approx 0.620735$, $b \approx 0.708194$ and $c \approx 0.591663$.

A comparison between the theoretic equations of state and the simulations is shown in figure 3.2. All simulations were performed on a cubic simulation box. The values for low densities ranged from 64 particles in a box of length 20 to $1,000$ particles in a box of length 21. For high densities, the number of particles was fixed at 108 and the box length varied from 9.65 to 8.865. When simulating high densities, we place particles initially on an FCC lattice. An example of a simulation of spheres is shown in figure 3.1.

The results of the simulations agree with the described theoretical equations of state. With the simulation of spheres working, it could be expanded to simulate non-spherical particles as well.

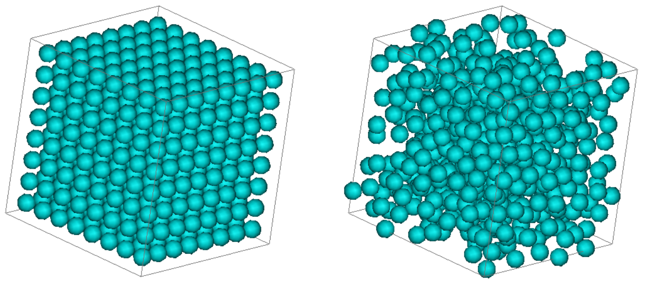


Figure 3.1: An example of a simulation of spheres with the first (cubic) and last configuration.

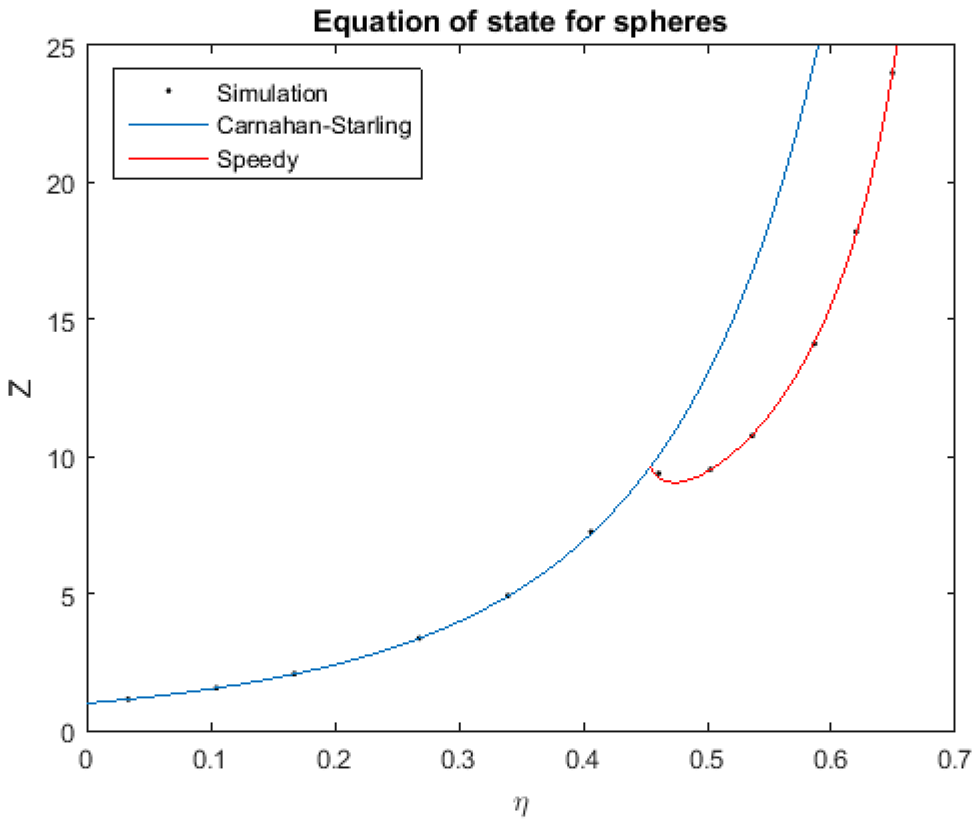


Figure 3.2: The compressibility factor $Z = \frac{PV}{Nk_BT}$ in a system of hard spheres is plotted against the packing fraction $\eta = \frac{Nv_0}{V}$ of the system. The red line shows the Carnahan-Starling equation of state, which is reliable at low packing fractions, and the blue line shows the Speedy equation of state, which is reliable at high packing fractions. The black dots are values obtained by performing EDMD simulations and correspond well to the equations.

3.1.2 Polyhedra

After establishing the correctness of our simulation for spheres, we adapted it to polyhedra.

An example of a simulation of the simplest polyhedra, namely triangles, is shown in figure 3.3.

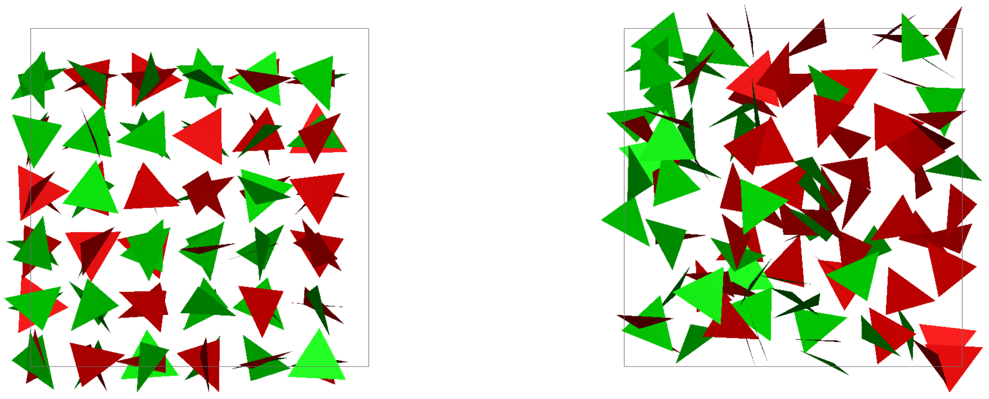


Figure 3.3: An example of a simulation of triangles with the first and last configuration. The red and green colours indicate the two sides of a triangle.

With the program working for triangles, we could construct three-dimensional objects consisting of triangles. We start with the simplest shape, a tetrahedron. Just like for spheres, one can determine the equation of state of tetrahedra from simulations. An example of a simulation of tetrahedra at low density is shown in figure 3.4. The final configuration for the highest density, still in the fluid phase, is shown in figure 3.5. The obtained equation of state is shown in figure 3.6.

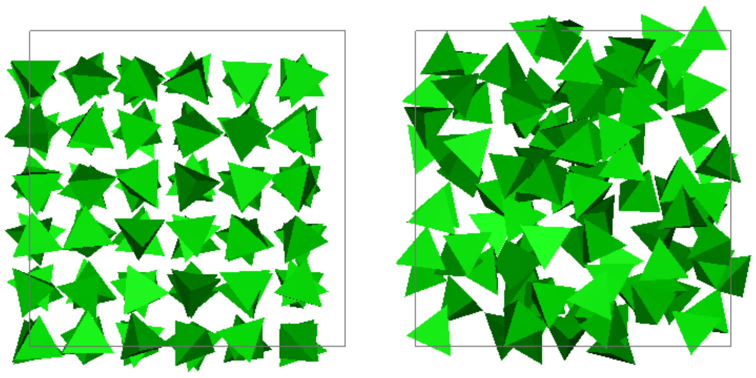


Figure 3.4: An example of a simulation of tetrahedra with the first and last configuration.

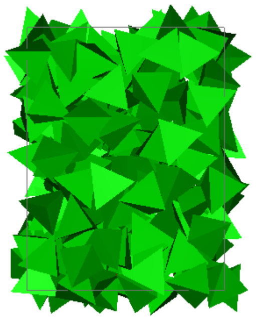


Figure 3.5: The final configuration for the highest density studied. The system is still in the fluid phase.

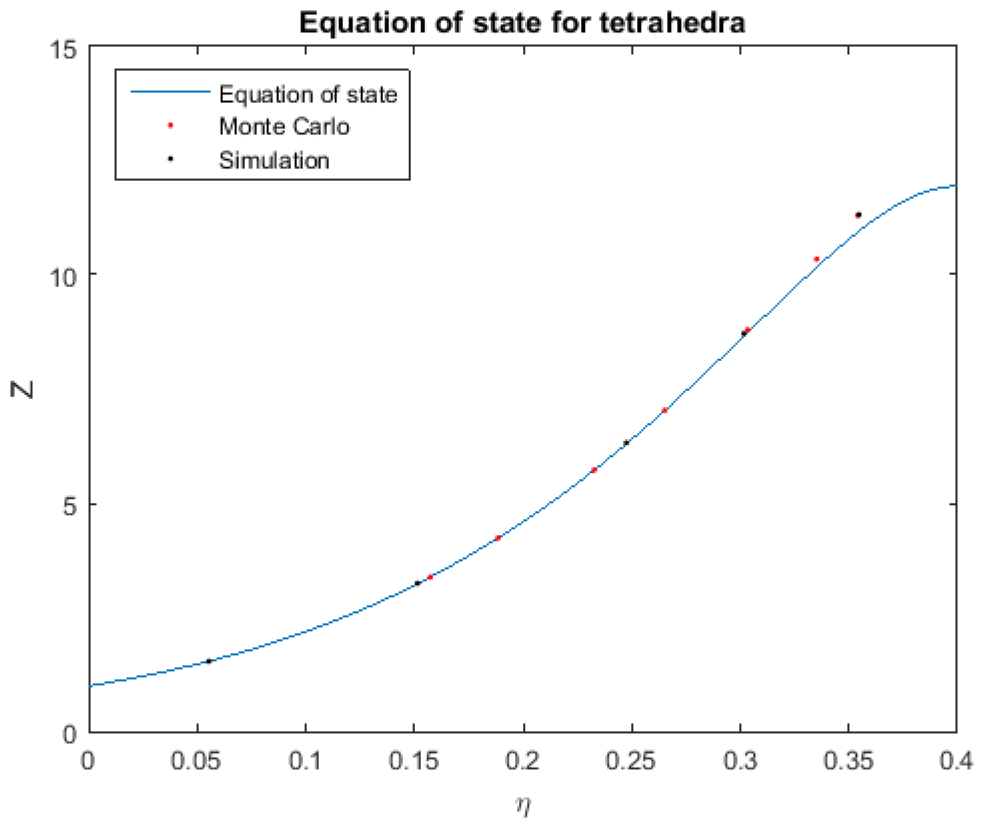


Figure 3.6: The compressibility factor $Z = \frac{PV}{Nk_BT}$ *in a system of hard tetrahedra is plotted against the packing fraction* $\eta = \frac{Nv_0}{V}$ *of the system. The red dots are the result of Monte Carlo equations from [20], the blue line is the derived equation of state from the same source. (These results from literature agree with those in [18].) The black dots are the values obtained by performing EDMD simulations and correspond well to the Monte Carlo equations: in particular, underneath the black point at the highest packing fraction, there is a red point.*

3.1.3 Twisted triangular prisms

The particles we are interested in are so-called twisted triangular prisms as described in [9], where the particle shape is based on parameters h, ω, α, as shown in figure 3.7.

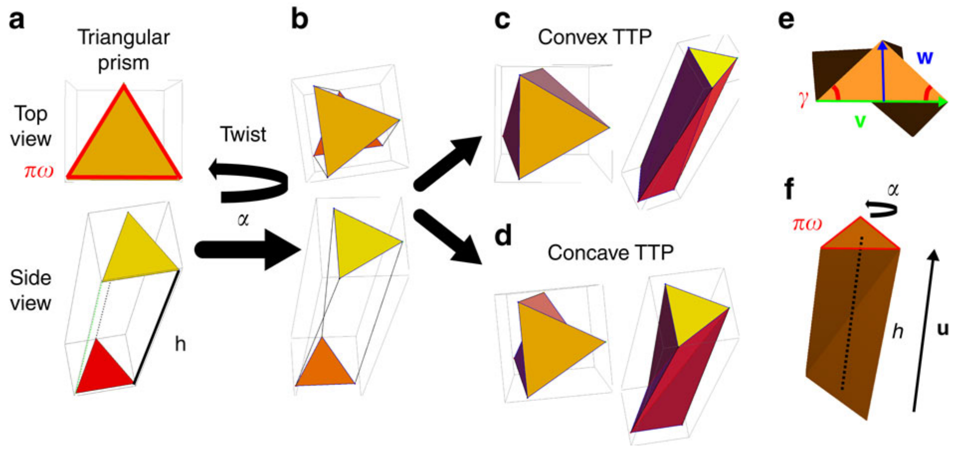


Figure 3.7: "A twisted triangular prism (TTP) is constructed from an elongated prism of height h with (isosceles) triangular bases, determined by the angle γ, and perimeter $\pi\omega$. The width ω is used as the unit of length. To introduce chirality, one triangular base is twisted by an angle α relative to the other one and additional edges are constructed to obtain flat faces." Souce: [9].

To test the program, we determined the equation of state of these particles for $h/\omega = 5$ and $\alpha = 0.7$. For packing fractions between 0 and 0.15, a correct starting configuration could still be obtained by hand. For higher densities, Monte Carlo simulations were performed in the Soft Condensed Matter group at the Utrecht University to provide suitable starting configurations for the EDMD simulations. The simulations give the equation of state in figure 3.9. At low densities, the figure

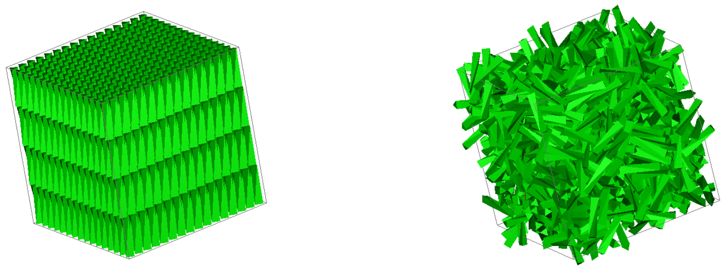


Figure 3.8: Twisted triangular prisms at packing fraction 0.15 at the starting configuration and after 1,500,000 collisions.

shows results of simulations with both normal and twisted periodic boundary conditions. One can see that the type of periodic boundary conditions does not influence the pressure, because the system is still in the fluid phase.

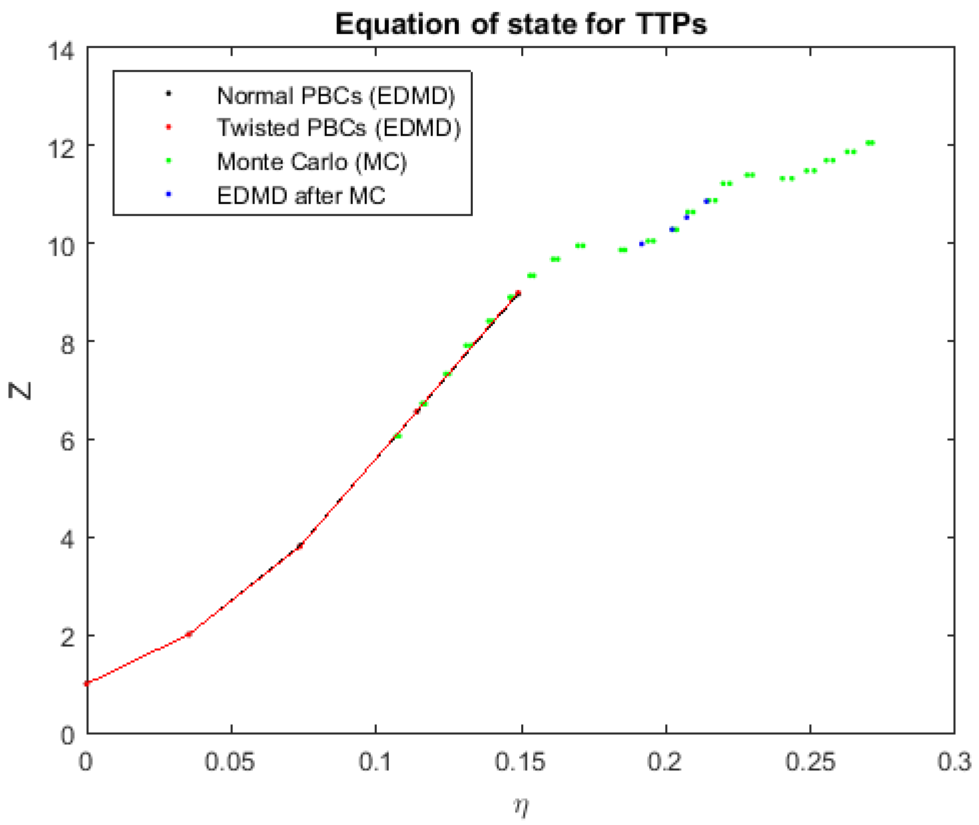


Figure 3.9: The compressibility factor $Z = \frac{PV}{Nk_BT}$ *in a system of hard twisted triangular prisms is plotted against the packing fraction* $\eta = \frac{Nv_0}{V}$ *of the system. The black dots were obtained by performing EDMD simulations with normal periodic boundary conditions, red points are with twisted periodic boundary conditions with the red line as an estimate. The green points are the results of performing Monte Carlo simulations. The blue points were obtained by taking configurations from these Monte Carlo simulations as starting point for EDMD simulations with normal periodic boundary conditions. The results from both types of simulations, Monte Carlo and EDMD, agree with each other.*

3.1.4 Technical remark

We would like to mention here that the stability of the program described in this chapter depends largely on the particle shape. For very elongated particles, the rotational velocity can be large, which can lead to many iteration steps in the collision prediction algorithm. Because of accumulating rounding errors, even with the use of double precision, a collision may be predicted incorrectly. This means that a long simulation of twisted triangular prisms may need many restarts, while a similar simulation of tetrahedra does not. An option to avoid restarts would be to round down numbers in the collision prediction instead of rounding them to the nearest value, but this could lead to even more iteration steps.

3.2 Efficiency

The running times of the different simulations described so far depend on many factors. These include the number of particles, the density, the particle shape, the phase, the temperature, the number of collisions, etcetera. Therefore, the speed-up obtained by using, for example, an event tree, is not uniquely defined for all systems. Even so, we would like to give an indication of the running times involved in the systems we have studied. The numbers should be taken as an illustration of the qualitative influence of different factors on the running time of the program.

The simulations were run on two different systems. The first one was a personal computer, the characteristics of which are shown below. The second one was the Batch system of the supercomputer Cartesius at surfSARA in Amsterdam. (Only thin nodes were used.) Memory is defined as the amount of memory per node.

System	**Nodes**	**Cores**	**Clock (GHz)**	**Memory (GB)**
PC	1	2	2.66	72
Cartesius	1,080	24	2.60	64
	540	24	2.40	64

Table 3.1: Characteristics of the computer clusters on which the simulations in this section were run.

3.2.1 Spherical particles

We begin with simulations for the simplest shape we studied, the spherical particles. These simulations were done on the personal computer. We consider five different versions of the program. For each new version of the program, an effort is made to make it more efficient compared to the previous version.

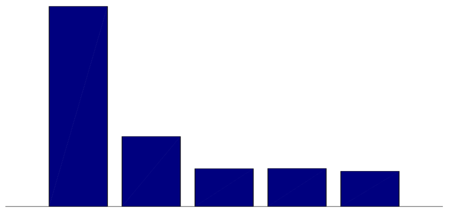
The first version of the program we call 'Array of $\mathcal{O}(N^2)$'. In this program, all predicted collisions between particles are stored in an $N \times (N+1)$ array A, where N is the number of particles. For all $0 \leq i < j < N$, the predicted time of collision between particle i and j is stored in $A(i,j)$. The minimum of all entries in row i of matrix A is stored in $A(i,N)$. These minima are in turn used to calculate the smallest entry in the entire array. In other words, determining the smallest time of collision is $\mathcal{O}(N^2)$.

The second version of the program is 'Event tree', which uses the event tree described earlier. This program already uses the cell list, where particle collisions are only calculated between particles in the same or neighbouring cells. From this program, only local times were added to the particles to obtain 'Local times', which is the third version of our program.

In the fourth version of the program, 'Cell crossing', the previously calculated collisions of a particle are re-used as much as possible when a particle crosses from one cell to the next, instead of being re-calculated. Finally, in the fifth version, 'List of $\mathcal{O}(1)$, the event tree of order one as described earlier is used.

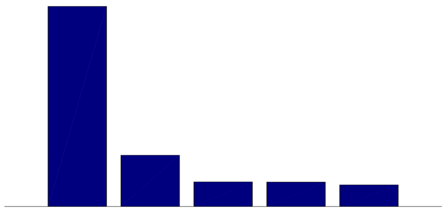
Typical running times are in the following tables.

System: 512 particles, $20 \times 20 \times 20$ box, 1,000,000 particle collisions.



Program	Time (s)	Relative
Array of $\mathcal{O}(N^2)$	103.0	100.0
Event tree	36.0	35.0
Local times	19.5	18.9
Cell crossing	19.6	19.0
List of $\mathcal{O}(1)$	18.1	17.6

System: 1,000 particles, $21 \times 21 \times 21$ box, 1,000,000 particle collisions.



Program	Time (s)	Relative
Array of $\mathcal{O}(N^2)$	205.2	100.0
Event tree	52.5	25.6
Local times	25.3	12.3
Cell crossing	25.1	12.2
List of $\mathcal{O}(1)$	22.2	10.8

From the tables, we conclude that the speed-up associated with the different modifications to the program is larger if the number of particles is higher and/or the density is higher, which is what we would expect. Also, for a very small number of spherical particles, re-using previously calculated collisions does not result in a speed-up, because the collision formula is so cheap.

3.2.2 Different shapes

Here, we use the last version of the program described in the previous subsection on the same personal computer. The twisted triangular prisms have $h/\omega = 5$, $\alpha = 0.7$, and an equilateral base.

System: 512 particles, $20 \times 20 \times 20$ box, 10,000 particle collisions.

Program	Time (s)	Relative
Spheres	0.2	1.0
Triangles	12.7	63.5
Tetrahedra	75.5	377.5
TTPs	243.9	1,219.5

System: 512 particles, $20 \times 20 \times 20$ box, 100,000 particle collisions.

Program	Time (s)	Relative
Spheres	1.8	1.0
Triangles	125.8	69.9
Tetrahedra	765.1	425.1
TTPs	2,509.8	1,394.3

At this density, the twisted triangular prisms (TTPs) are still in the fluid phase, so nearest neighbour lists do not improve the running time. (For example, 10,000 collisions of the above system for TTPs takes 486.7 s.) Also, the running time can become worse as the simulation continues, depending on the system. It is clear however, that the program has bad scalability in the number of triangles per particle.

3.2.3 Nearest neighbour lists

These simulations were performed on Cartesius. The system consists of twisted triangular prisms with $h/\omega = 5$, $\alpha = 0.7$ and an (almost) equilateral base, a reduced temperature of $k_BT = 0.2$, and a fixed time in the simulation of 20 seconds.

System: 2,400 particles, $\gamma = 1.04$, $29.748473 \times 21.798350 \times 22.568546$ box, $\sim$ 500,000 particle collisions.

Program	Time (s)	Relative
Without NNLs	118,472.2	100.0
With NNLs	44,833.9	37.8

System: 1,024 particles, equilateral base, $8.0 \times 8.0 \times 8.0$ box, $\sim$ 315,000 particle collisions.

Program	Time (s)	Relative
Without NNLs	52,659.4	100.0
With NNLs	24,888.3	47.3

The tables show that the speed-up for nearest neighbour lists depends on the system. For most of our systems, the speed-up was between 2.0 and 3.0.

3.2.4 Parallelization

These simulations were done on Cartesius. The system consists of twisted triangular prisms with $h/\omega = 5$, $\alpha = 0.7$ and an equilateral base, a reduced temperature of $k_BT = 0.2$, and a fixed time in the simulation of 10 seconds. No nearest neighbour lists were used. Note that the parallel program using only one processor is slightly different from the sequential algorithm, since changes for parallelization are still performed.

System: 1,024 particles, equilateral base, $8.0 \times 8.0 \times 8.0$ box, $\sim$ 150,000 particle collisions.

Number of processors	Time (s)	Relative
Sequential	25,426.3	100.0
1	25,321.3	99.6
2	20,756.6	81.6
4	17,834.7	70.1
8	15,731.0	61.9

Table 3.2: Without nearest neighbour lists.

For the system we studied, both parallelization and nearest neighbour lists are possible. Hence, we wrote a program using both, and performed the same tests as before.

Number of processors	Time (s)	Relative
Sequential	11,500.1	100.0
1	11,634.9	101.2
2	10,962.4	95.3
4	11,116.0	96.7
8	11,148.2	96.9

Table 3.3: With nearest neighbour lists.

We conclude that for the system already using nearest neighbour lists, the speed-up from parallelization is modest. The reason is that parallelization divides the predictions of collisions over multiple processors, but nearest neighbour lists already reduces the number of collisions that has to be predicted. The predictions that remain are practically always on the current processor, which makes the advantage of parallelization modest.

Note however that this is only the case for very elongated particles. Since this is the system we are interested in, the current parallelization is for us not beneficial, but it could be for systems of more spherical particles. An indication of this is shown below.

System: 847 particles, equilateral base, $7.25 \times 7.25 \times 8$ box, $h/\omega = 3.0$, $\sim$ 60,000 particle collisions.

Number of processors	Time (s)	Time NNLs (s)
Sequential	5,649.7	4,097.5
1	5,316.0	3,819.2
2	4,527.1	3,624.0
4	3,844.1	3,367.9
8	3,367.3	3,212.0

So the parallel algorithm with 8 processors using NNLs is 21.6 per cent faster than the same sequential algorithm.

3.2.5 Discussion

The simulations described in this chapter show the following trend. A simulation of particles can be improved significantly by using an $\mathcal{O}(1)$ event tree with a cell list and local times and by re-using previously calculated collisions at a cell crossing. Adapting such a simulation to particles of which the surface consists of triangles drastically increases the running time, especially for particles consisting of many triangles. For very elongated particles at a high density, the running time can be improved again by using nearest neighbour lists, but the simulation time is still nowhere near that of spherical particles anymore.

According to literature, it might be better to only have one collision in the tree for each particle. In this case, a Complete Binary Tree can be used. However, this makes the program more difficult to extend with different optimizations.

The distance between two objects could also be found by using a different algorithm for determining the distance between the particles. An algorithm that only works for convex parts is the so-called Gilbert-Johnson-Keerthi (GJK) algorithm [33], which is an iterative algorithm. This algorithm could be applied to the different convex parts of the particles. Whether this results in a speed-up or not depends on the number of convex parts needed.

Parallelizing the program has both advantages and disadvantages. One advantage is that the algorithm described so far still has room for improvement, which could lead to bigger speed-ups than those reported here. Also, the parallelization could give an improvement for non-elongated particles as well, or for elongated particles whose neighbours change rapidly, in contrast to the nearest neighbour lists. Finally, for a large number of particles, the required amount of memory is distributed over the processors, which could also improve the use of Cache. (For the systems we studied, a maximum of 2,520 particles was used, for which the use of Cache is not a limiting factor yet.)

On the other hand, we think it is safe to say that parallelization is significantly more programming work than implementing nearest neighbour lists. This is partly because parallelization also complicates practical issues like keeping track of the global pressure, restarting a simulation if one processor fails, etcetera. The reader will have to be the judge whether the obtained speed-up weighs up against the extra programming time.

For the simulations described in the next section, we used the fastest sequential program, which contained nearest neighbour lists. The simulations were performed on two different computer clusters of the Utrecht University, the first half of each simulation on Mars and after that the second half on Thor.

System	Nodes	Cores	Clock (GHz)	Memory (GB)
Mars	15	8	2.0	4
	14	4	2.2	2
	13	4	2.0	2
	6	4	2.2	4
Thor	8	48	2.8	128
	7	48	2.6	128

Table 3.4: Characteristics of the computer clusters on which the simulations in the next section were run.

3.3 Cholesteric liquid crystals

To determine the order of magnitude of the torque differences to be studied, we first ran a simulation on a small configuration of 396 twisted triangular prisms with $h/\omega = 5$, $\alpha = 0.7$ and $\gamma = 0.75$ under normal periodic boundary conditions. The packing fraction was $\eta = 0.14877$ and the compressibility factor $Z = 9.2333$. After 1,460,000 collisions, the torque matrix was the following.

$$\Pi_{\alpha\beta} = \begin{bmatrix} 0.009472 & 0.000925 & 0.001850 \\ 0.000323 & -0.024452 & 0.000141 \\ -0.003105 & 0.002619 & -0.023422 \end{bmatrix}$$

The system was in a cholesteric liquid crystal phase, as shown on the right. According to theory, the off-diagonal entries of this matrix should be zero, so the result is only accurate up to the third decimal. The absolute size of the torque of interest is quite small compared to the variation within the simulation, which complicates a precise calculation.

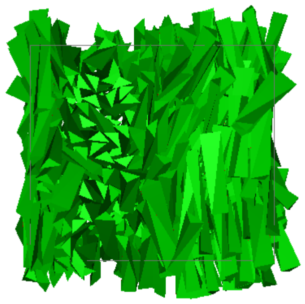
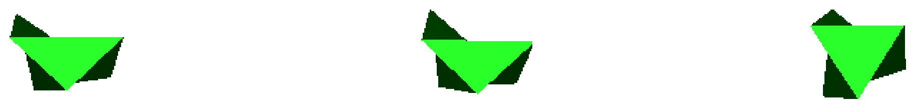


Figure 3.10: A system of 396 *particles with* $h/\omega = 5$, $\alpha = 0.7$, $\gamma = 0.75$, $\eta = 0.14877$ *and* $Z = 9.2333$ *in a cholesteric phase.*

Next, we ran simulations on three different particles. All particles were twisted triangular prisms, but the parameters determining their shape differed.

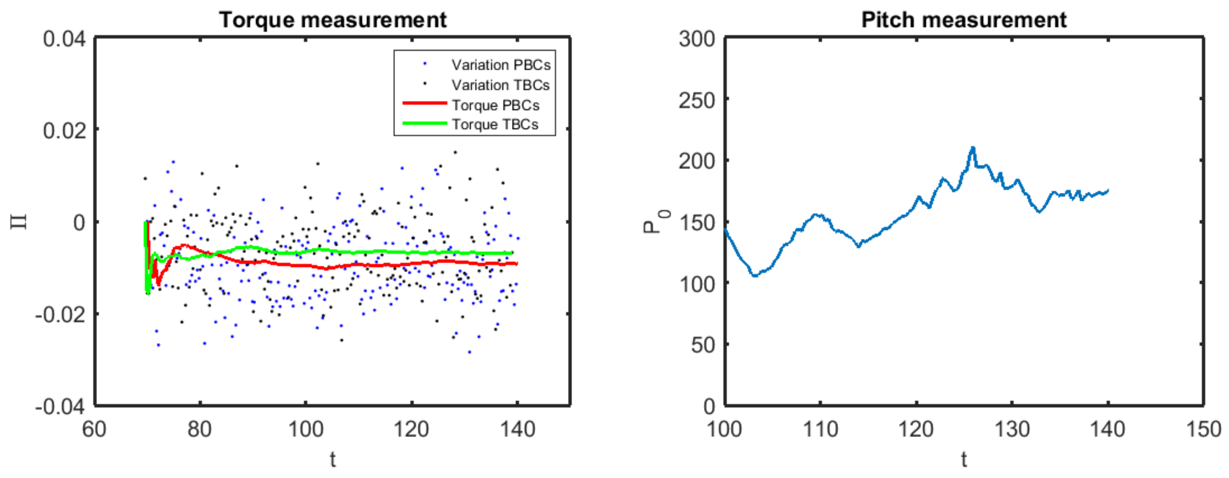


(a) $\alpha = 0.6$, $\gamma = 0.75$ *(b)* $\alpha = 0.7$, $\gamma = 0.75$ *(c)* $\alpha = 0.7$, $\gamma = 1.0$

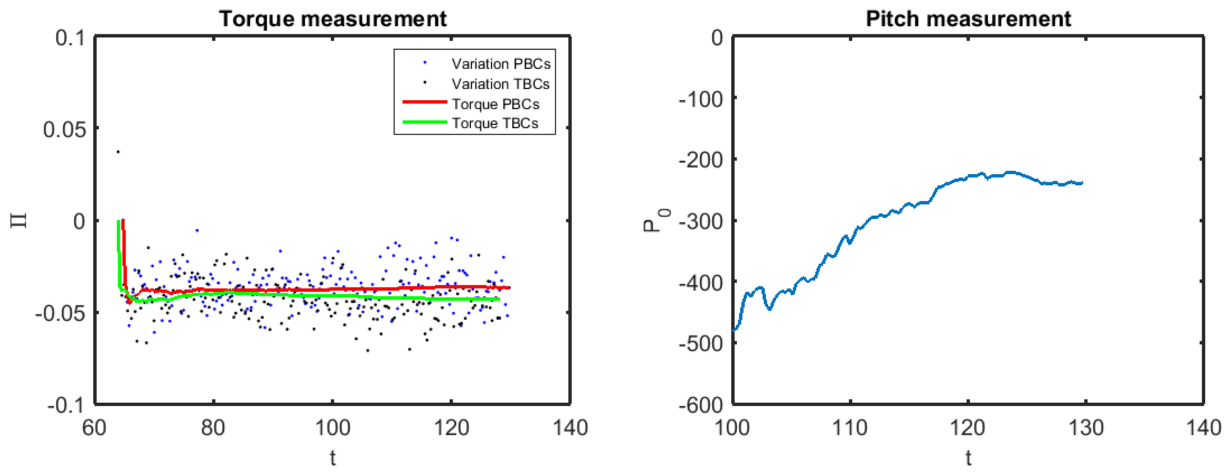
Figure 3.11: The three different twisted triangular prisms on which we performed torque measurements. The second particle is twisted slightly more than the first, and the third particle has a larger base than the second.

The systems we studied consisted of 2,520 particles. For each configuration, we performed two simulations: one with normal periodic boundary conditions, and one with twisted periodic boundary conditions in one direction. Both systems were equilibrated over 2,000,000 collisions, after which the torque was measured in the consecutive collisions. Apart from measuring the torque over many (at least 1,000,000) consecutive collisions, we also measured the torque over time intervals containing 10,000 collisions separately. We will denote this as the variation of the torque.

The results are the following. The time is in arbitrary units depending on the temperature of the system.

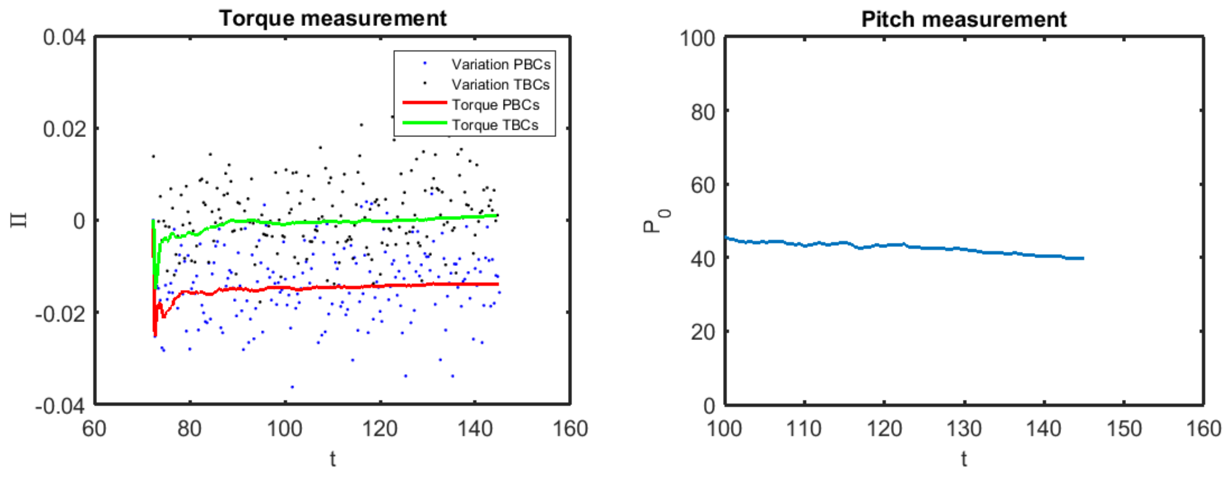


(a) *Packing fraction* $\eta = 0.20825$*, compressibility factor* $Z = 10.833$*.*

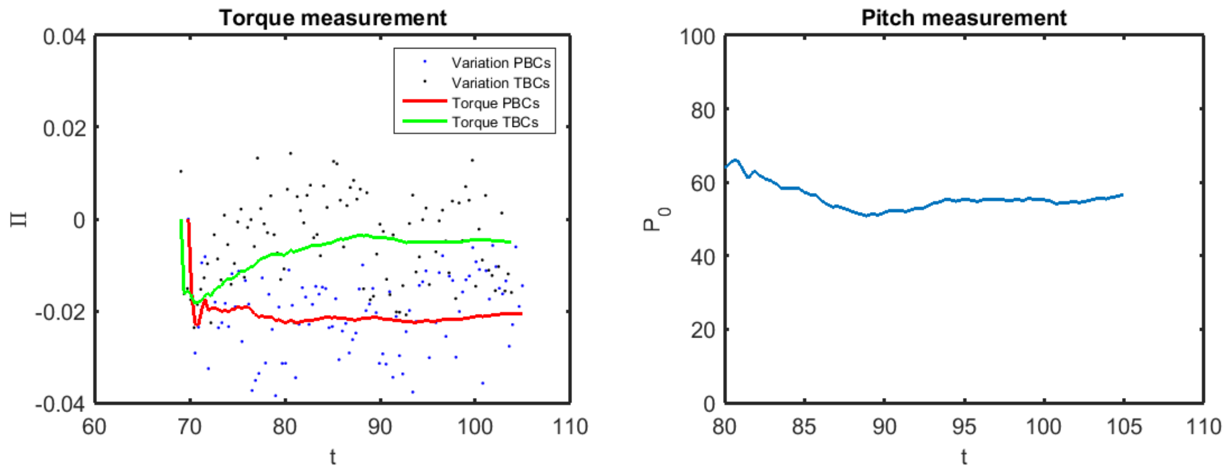


(b) *Packing fraction* $\eta = 0.22528$*, compressibility factor* $Z = 11.327$*.*

Figure 3.12: The figures show the direct measurement of the pitch $\mathcal{P}_0$ *of a cholesteric liquid crystal in simulations at two different packing fractions. Given a system, two simulations are run, one with normal (PBCs) and one with twisted (TBCs) periodic boundary conditions. The left figures show a measurement of the torque* Π *in the system over time, where the variation is given by the instantaneous torque. The right figures show the resulting measurement of the pitch* $\mathcal{P}_0$*. The cholesteric pitch is expressed in the length of the radius of a sphere enclosing one particle. The system consists of* $2,520$ *twisted triangular prisms with* $\alpha = 0.6$*,* $h/\omega = 5$*,* $\gamma = 0.75$*. After* 2 *million collisions of equilibration, the measurement was done over another* 2 *million collisions.*

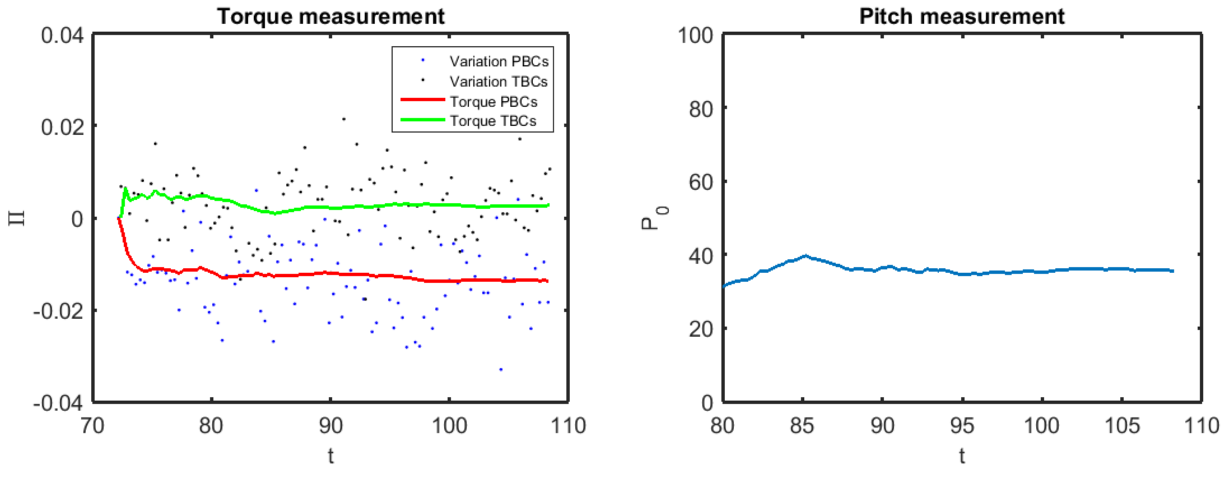


(a) *Packing fraction* $\eta = 0.17583$, *compressibility factor* $Z = 10.547$.

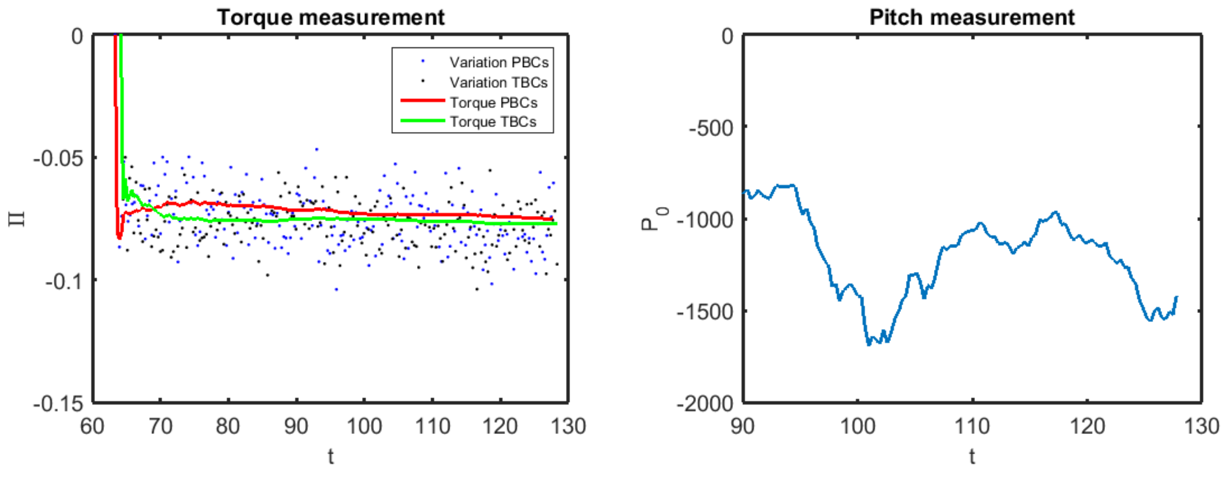


(b) *Packing fraction* $\eta = 0.18115$, *compressibility factor* $Z = 10.775$.

Figure 3.13: The figures show the direct measurement of the pitch $\mathcal{P}_0$ *of a cholesteric liquid crystal in simulations at two different packing fractions. Given a system, two simulations are run, one with normal (PBCs) and one with twisted (TBCs) periodic boundary conditions. The left figures show a measurement of the torque* Π *in the system over time, where the variation is given by the instantaneous torque. The right figures show the resulting measurement of the pitch* $\mathcal{P}_0$. *The cholesteric pitch is expressed in the length of the radius of a sphere enclosing one particle. The system consists of* $2,520$ *twisted triangular prisms with* $\alpha = 0.7$, $h/\omega = 5$, $\gamma = 0.75$. *After* 2 *million collisions of equilibration, the measurement was done over another* 2 *million collisions for figure (a) or* 1 *million collisions for figure (b).*



(a) Packing fraction $\eta = 0.22158$, compressibility factor $Z = 11.369$.



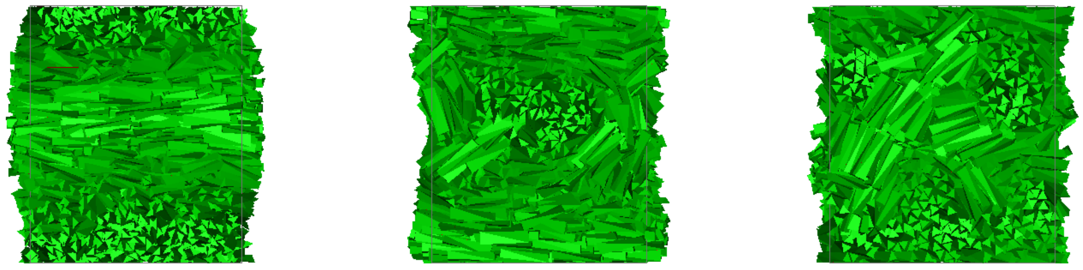
(b) Packing fraction $\eta = 0.25476$, compressibility factor $Z = 11.940$.

Figure 3.14: The figures show the direct measurement of the pitch $\mathcal{P}_0$ of a cholesteric liquid crystal in simulations at two different packing fractions. Given a system, two simulations are run, one with normal (PBCs) and one with twisted (TBCs) periodic boundary conditions. The left figures show a measurement of the torque Π in the system over time, where the variation is given by the instantaneous torque. The right figures show the resulting measurement of the pitch $\mathcal{P}_0$. The cholesteric pitch is expressed in the length of the radius of a sphere enclosing one particle. The system consists of $2,520$ twisted triangular prisms with $\alpha = 0.7$, $h/\omega = 5$, $\gamma = 1.0$. After 2 million collisions of equilibration, the measurement was done over another 1 million collisions for figure (a) or 2 million collisions for figure (b).

We note that the torque for the system with twisted periodic boundary conditions is indeed different from the torque for the system with normal periodic boundary conditions. However, this difference is relatively small both in absolute value and when compared to the variation of the torque. Because the formula for the cholesteric pitch uses a division by this difference, the value of $\mathcal{P}_0$ can blow up, as can be seen in figure 3.14b and in the beginning of figure 3.12b. By looking at the configurations, we see that these are the configurations which show defects instead of a smooth cholesteric twist, which do not disappear during the simulations. The initial configurations are shown in figure 3.15.

An example of the simulation behaviour is shown in figure 3.17. Despite the long EDMD simulations, the final configurations show a system that is not relaxed yet. For all higher densities, we also expect the torque measurement method to fail, since neither the normal nor the twisted periodic boundary conditions allow the equilibration of the system.

Even when the measured pitch does not blow up, a small difference in the measured torque can give a large difference in the resulting pitch. This can be seen in figure 3.12a, where $\mathcal{P}_0$ varies considerably over time.



$\alpha = 0.6$, $\gamma = 0.75$, $\eta = 0.20825$
See also figure 3.12a.

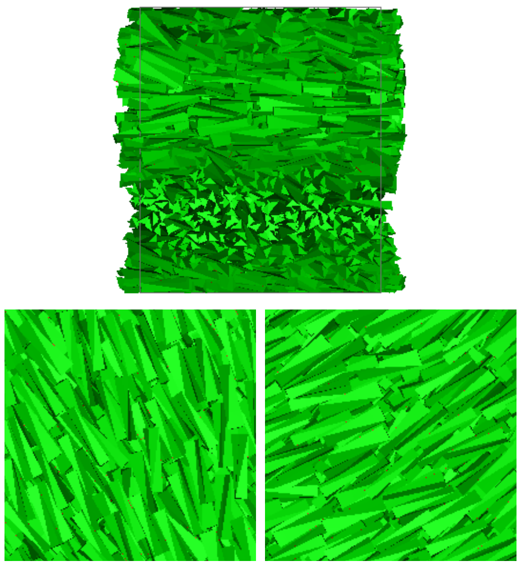
$\alpha = 0.6$, $\gamma = 0.75$, $\eta = 0.22528$
See also figure 3.12b.

$\alpha = 0.7$, $\gamma = 1.0$, $\eta = 0.25476$
See also figure 3.14b.

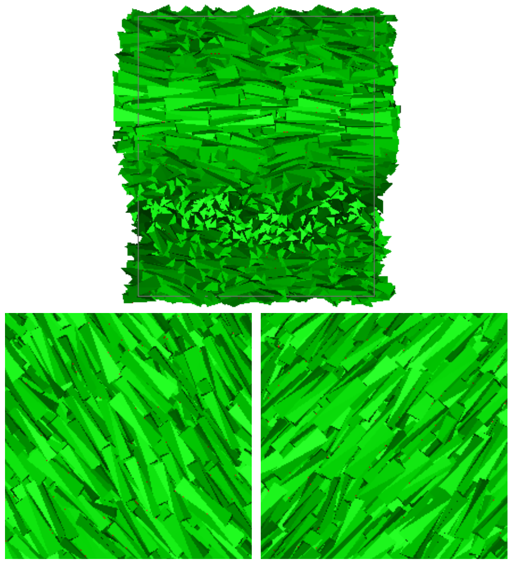
Figure 3.15: Some of the initial configurations of the simulations measuring the equilibrium pitch of a cholesteric liquid crystal. Note that the two configurations on the right contain defects, whereas the left configuration is a smooth cholesteric twist.

Finally, the pitch is positive and relatively constant in figures 3.13a, 3.13b and 3.14a. According to these figures, $\mathcal{P}_0$ is between 40 and 80, and a slight difference in the pressure (i.e., the volume) can give quite a difference in $\mathcal{P}_0$. An example of the simulation behaviour is shown in figure 3.16. Notice that for twisted periodic boundary conditions, the particle orientations at the top and bottom of the system are practically perpendicular, which means that they fit on top of each other when taking the twist in the boundary conditions into account. This suggests that the system can indeed reach an equilibrated structure in a long EDMD simulation.

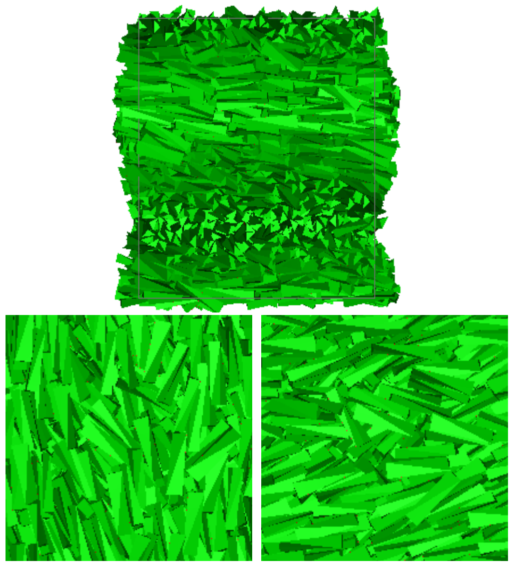
Although the order of the measured pitches is correct here, its seems that longer simulations are needed to accurately determine the value of the equilibrium pitch.



(a) Starting configuration

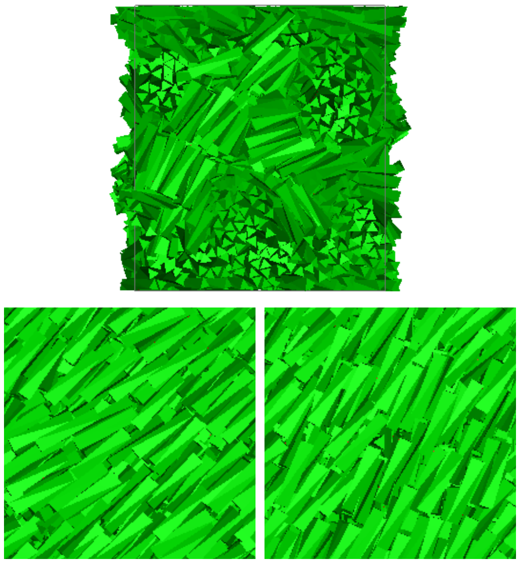


(b) Normal periodic boundary conditions

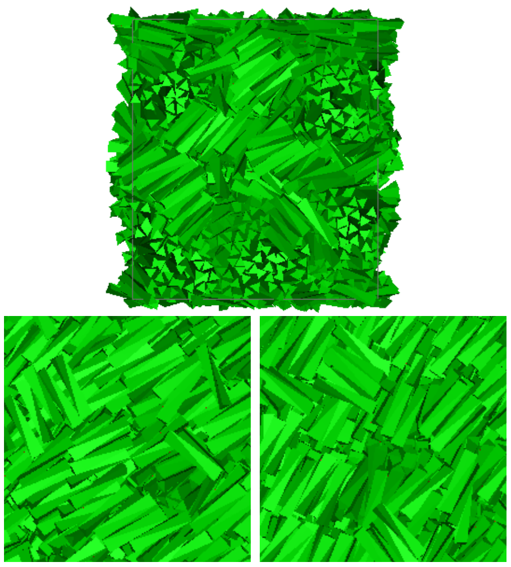


(c) Twisted periodic boundary conditions

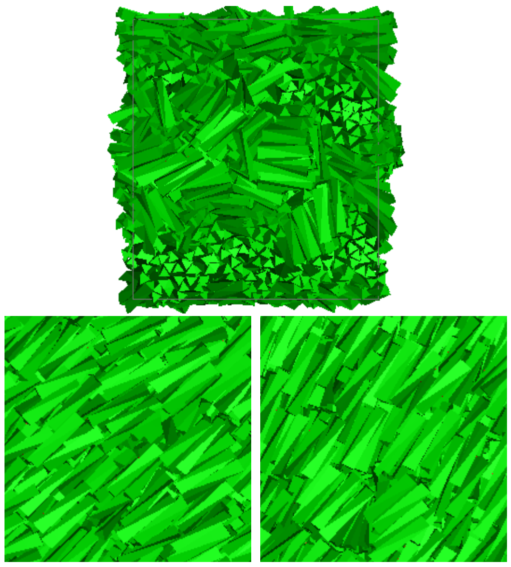
Figure 3.16: The path of one configuration with a successful pitch measurement. The three figures correspond to a front view (top), a top view (left below) and a bottom view (right below). This is a system of $2,520$ *twisted triangular prisms with* $\alpha = 0.7$, $h/\omega = 5$, $\gamma = 0.75$. *After* 2 *million collisions of equilibration, the measurement was done over another* 2 *million collisions.*



(a) Starting configuration



(b) Normal periodic boundary conditions



(c) Twisted periodic boundary conditions

Figure 3.17: The path of one configuration with a failed pitch measurement. The three figures correspond to a front view (top), a top view (left below) and a bottom view (right below). Neither normal nor twisted periodic boundary conditions result in a structured cholesteric phase. This is a system of $2,520$ *twisted triangular prisms with* $\alpha = 0.7$, $h/\omega = 5$, $\gamma = 1.0$. *After* 2 *million collisions of equilibration, the measurement was done over another* 2 *million collisions.*

3.3.1 Discussion

The results from measuring the cholesteric pitch are not accurate enough yet. This could be because the equilibration time was not reached yet, or because the choice for a spherical moment of inertia increased the required simulation time. It could be better to use a moment of inertia which is at least elongated, for example that of an ellipsoid.

However, the computational time is already quite long, and would probably increase when using a different moment of inertia. On that note, it might be interesting to equilibrate the configuration with normal and twisted periodic boundary conditions using a Monte Carlo simulation, and then only measure the torque using an EDMD simulation (instead of using this for equilibration as well).

Chapter 4

Conclusion & Outlook

We have shown that the EDMD algorithm is not only an efficient method to study a system of spherical particles, but that it can be extended to work for polyhedral, not necessarily convex particles as well. The equations of state of spheres, tetrahedra and even twisted triangular prisms could be reproduced accurately. By using Monte Carlo simulations to produce the starting configurations, we were able to simulate very dense systems of twisted triangular prisms, for which a cholesteric liquid crystal phase occurs. The running time for this system was improved on by using an event tree, a cell list, local times, bounding spheres and oriented bounding boxes.

The algorithm was adapted to accommodate both normal and twisted periodic boundary conditions. This allowed a torque measurement which can be used to measure the cholesteric pitch of a system directly. For configurations with defects, the measurements were not reliable, and in some cases, the small absolute value of the torque measurement made the resulting pitch blow up. However, some simulations indicate that measuring the pitch directly this way is indeed possible, and could give an accurate result.

Despite the many methods used to reduce the running time of the program, it still turned out to be an important aspect of this project. By using the distance between two triangles on polyhedra instead of between spheres, the program slowed down much more than anticipated. Moreover, for the systems of twisted triangular prisms we studied, the time to handle a fixed number of collisions increases as the simulation progresses. Because of the long shape of these particles, there is a lot of rotational energy in the system, which does not equilibrate the configuration very well. As an indication, the simulations described in the previous section took multiple weeks.

The algorithm can be improved on in several ways. In general, storing less events could give an improvement on the time associated with actions in the event tree. For polyhedra, maybe the algorithm for determining the distance between two triangles can be made more efficient. Another option is to determine the distance between two polyhedra by using an algorithm called the Gilbert-Johnson-Keerthi algorithm, or GJK for short. Because of the iterative nature of this algorithm, we were inclined to use the exact approach. However, the accuracy of the distance between two particles is less important in the simulation than the efficiency, since a fast program can simulate many collisions and hence give a more accurate torque measurement. Since we are already using nearest neighbour lists, the improvement by using a parallel algorithm would be modest, but for different systems, this could be an interesting option too.

Once the running time of the program has been improved upon, it would be interesting to do more simulations to see if the results observed so far are consistent for other systems as well. If the running time cannot be improved much, it could be worthwhile to equilibrate the configurations using Monte Carlo simulations with both normal and twisted periodic boundary conditions, and then only do the torque measurement in EDMD.

In terms of the actual pitch measurement, this could become more accurate if the simulations are longer. If this is not the case, then possibly the assumption underlying the use of a spherical moment of inertia is wrong, and it would be better to use an ellipsoidal moment of inertia or even the "real" moment of inertia based on the actual particle shape.

In Monte Carlo simulations with walls, systems have been observed which have a twist in two directions (but not at the same time). It would be interesting to see under which conditions they are stable by using these torque measurements. This is something that we are still investigating.

Bibliography

[1] Alder, B. J., and Wainwright, T. Studies in molecular dynamics. I. general method. *The Journal of Chemical Physics 31*, 2 (1959), 459–466.

[2] Allen, M. P., and Masters, A. J. Computer simulation of a twisted nematic liquid crystal. *Molecular Physics 79*, 2 (1993), 277–289.

[3] Bannerman, M. N., Lue, L., and Woodcock, L. V. Thermodynamic pressures for hard spheres and closed-virial equation-of-state. *The Journal of chemical physics 132*, 8 (2010), 084507.

[4] Bernard, E. P., and Krauth, W. Two-step melting in two dimensions: First-order liquid-hexatic transition. *Physical review letters 107*, 15 (2011), 155704.

[5] Catto, E. Physics for game programmers; continuous collision. http://cacs.usc.edu/education/cs596/01-1LinkedListCell.pdf.

[6] de la Peña, L. H., van Zon, R., Schofield, J., and Opps, S. B. Discontinuous molecular dynamics for semiflexible and rigid bodies. *The Journal of chemical physics 126*, 7 (2007), 074105.

[7] Donev, A., Cisse, I., Sachs, D., Variano, E. A., Stillinger, F. H., Connelly, R., Torquato, S., and Chaikin, P. M. Improving the density of jammed disordered packings using ellipsoids. *Science 303*, 5660 (2004), 990–993.

[8] Donev, A., Torquato, S., and Stillinger, F. H. Neighbor list collision-driven molecular dynamics simulation for nonspherical hard particles. I. algorithmic details. *Journal of Computational Physics 202*, 2 (2005), 737–764.

[9] Dussi, S., and Dijkstra, M. Entropy-driven formation of chiral nematic phases by computer simulations. *Nature Communications 7* (2016).

[10] Eberly, D. Distance between point and triangle in 3d. *Magic Software, http://www. magic-software. com/Documentation/pt3tri3. pdf* (1999).

[11] Eberly, D. Polyhedral mass properties (revisited). *Geometric Tools, LLC, Tech. Rep* (2002).

[12] Frenkel, D. Simulations: The dark side. *The European Physical Journal Plus 128*, 1 (2013), 1–21.

[13] Frenkel, D., Lekkerkerker, H., and Stroobants, A. Thermodynamic stability of a smectic phase in a system of hard rods.

[14] Frenkel, D., and Maguire, J. Molecular dynamics study of the dynamical properties of an assembly of infinitely thin hard rods. *Molecular physics 49*, 3 (1983), 503–541.

[15] Frenkel, D., Mulder, B., and McTague, J. Phase diagram of a system of hard ellipsoids. *Physical review letters 52*, 4 (1984), 287.

[16] Germano, G., Allen, M. P., and Masters, A. J. Simultaneous calculation of the helical pitch and the twist elastic constant in chiral liquid crystals from intermolecular torques. *Journal of Chemical Physics 116*, 21 (2002), 9422–9430.

[17] Goll, C. Parallel algorithms for molecular dynamics simulations in fluids.

[18] Haji-Akbari, A., Engel, M., Keys, A. S., Zheng, X., Petschek, R. G., Palffy-Muhoray, P., and Glotzer, S. C. Disordered, quasicrystalline and crystalline phases of densely packed tetrahedra. *Nature 462*, 7274 (2009), 773–777.

[19] Kirk, D. *Graphics Gems III (IBM Version): Ibm Version.* Elsevier, 2012.

[20] Kolafa, J., and Labík, S. Virial coefficients and the equation of state of the hard tetrahedron fluid. *Molecular Physics 113*, 9-10 (2015), 1119–1123.

[21] Marechal, M., Zimmermann, U., and Löwen, H. Freezing of parallel hard cubes with rounded edges. *The Journal of chemical physics 136*, 14 (2012), 144506.

[22] Miller, S., and Luding, S. Event-driven molecular dynamics in parallel. *Journal of Computational Physics 193*, 1 (2004), 306–316.

[23] Millington, I. *Game physics engine development.* Morgan Kaufmann Publishers Amsterdam, 2007.

[24] Mirtich, B. Fast and accurate computation of polyhedral mass properties. *journal of graphics tools 1*, 2 (1996), 31–50.

[25] Paul, G. A complexity O(1) priority queue for event driven molecular dynamics simulations. *Journal of Computational Physics 221*, 2 (2007), 615–625.

[26] Rapaport, D. Molecular dynamics study of a polymer chain in solution. *The Journal of Chemical Physics 71*, 8 (1979), 3299–3303.

[27] Rapaport, D. C. *The art of molecular dynamics simulation.* Cambridge university press, 2004.

[28] Sigurgeirsson, H., Stuart, A., and Wan, W.-L. Algorithms for particle-field simulations with collisions. *Journal of Computational Physics 172*, 2 (2001), 766–807.

[29] Smallenburg, F., Filion, L., Marechal, M., and Dijkstra, M. Vacancy-stabilized crystalline order in hard cubes. *Proceedings of the National Academy of Sciences 109*, 44 (2012), 17886–17890.

[30] Smallenburg, F., and Sciortino, F. Liquids more stable than crystals in particles with limited valence and flexible bonds. *Nature Physics 9*, 9 (2013), 554–558.

[31] Stijnman, M., Bisseling, R., and Barkema, G. Partitioning 3d space for parallel many-particle simulations. *Computer Physics Communications 149*, 3 (2003), 121–134.

[32] Tang, M., Kim, Y. J., and Manocha, D. C^2A: controlled conservative advancement for continuous collision detection of polygonal models. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on* (2009), IEEE, pp. 849–854.

[33] van den Bergen, G. Collision detection in interactive 3d environments. 2004.

[34] Binary tree.
http://web.cecs.pdx.edu/ sheard/course/Cs163/Doc/FullvsComplete.html.
Last accessed: 2016-06-01.

[35] Cell lists.
https://en.wikipedia.org/wiki/Cell_lists.
Last accessed: 2016-06-01.

[36] Elastic collisions of balls.
http://blogs.msdn.com/b/faber/archive/2013/01/09/elastic-collisions-of-balls.aspx.
Last accessed: 2016-05-01.

[37] Linked-list cell molecular dynamics.
http://cacs.usc.edu/education/cs596/01-1LinkedListCell.pdf.
Last accessed: 2016-05-01.

[38] Periodic boundary conditions.
http://isaacs.sourceforge.net/phys/pbc.html.
Last accessed: 2016-06-01.

[39] Phase transitions in liquid crystals.
http://soft-matter.seas.harvard.edu/index.php/Phase_transitions_in_liquid_crystals.
Last accessed: 2016-05-01.

Appendix A

Collision resolution

Colliding velocities

Consider the collision of two particles i and j, with mass M_i and M_j, translational velocity $\vec{v}_i$ and $\vec{v}_j$, and rotational velocity $\vec{w}_i$ and $\vec{w}_j$. Let $\hat{n}$ be the unit vector orthogonal to the plane of collision. Let $\vec{r}_i$ be the vector from the center of mass of particle i to the point of contact on particle i, and similarly for particle j. The velocity vector of the point of contact on particle i in the direction of $\hat{n}$ is then given by

$$v_i = (\vec{v}_i + \vec{\omega}_i \times \vec{r}_i) \cdot \hat{n}, \tag{A.1}$$

and similarly

$$v_j = (\vec{v}_j + \vec{\omega}_j \times \vec{r}_j) \cdot \hat{n}. \tag{A.2}$$

Define u_i as the velocity of particle i associated with $\hat{n}$ after the collision, and similarly for u_j. The relation between v_i, v_j, u_i, and u_j is given by the so-called *coefficient of restitution.*

Coefficient of restitution

The laws for conservation of momentum and energy state that

$$\begin{cases} \text{Conservation of momentum:} & M_i v_i + M_j v_j = M_i u_i + M_j u_j, \\ \text{Conservation of kinetic energy:} & M_i v_i^2 + M_j v_j^2 = M_i u_i^2 + M_j u_j^2. \end{cases}$$

From conservation of momentum, it follows that

$$M_i(v_i - u_i) = M_j(u_j - v_j). \tag{A.3}$$

From conservation of kinetic energy, it follows that

$$M_i(v_i + u_i)(v_i - u_i) = M_j(u_j + v_j)(u_j - v_j). \tag{A.4}$$

Since both particles are moving, $v_i \neq u_i$ and $u_j \neq v_j$. Hence, we can divide equation (A.4) by equation (A.3) to obtain $v_i + u_i = u_j + v_j$, which means that

$$-\frac{u_i - u_j}{v_i - v_j} = 1. \tag{A.5}$$

We will denote this by saying that the coefficient of restitution of an elastic collision equals one.

Conservation of momentum

Just like $\vec{v_i}$ and $\vec{v_j}$ describe the translational velocities of particle i and j before the collision, let $\vec{v_i}^*$ and $\vec{v_j}^*$ describe the translational velocities of particle i and j after the collision, and similarly for $\vec{w}_i^*$ and $\vec{w}_j^*$. Conservation of linear momentum states that

$$M_i\vec{v}_i^* + M_j\vec{v}_j^* = M_i\vec{v}_i + M_j\vec{v}_j. \tag{A.6}$$

This means that there exists an $x \in \mathbb{R}$ such that

$$\vec{v}_i^* = \vec{v}_i + \frac{x}{M_i}\hat{n}, \tag{A.7}$$

$$\vec{v}_j^* = \vec{v}_j - \frac{x}{M_j}\hat{n}. \tag{A.8}$$

The change in angular momentum of a particle is given by the so-called *torque* exerted on that particle, which gives conservation of angular momentum of the system. In this case, if I_i and I_j are the moments of inertia of particles i and j,

$$\vec{w}_i^* = \vec{w}_i + I_i^{-1}(\vec{r}_i \times (x\hat{n})), \tag{A.9}$$

$$\vec{w}_j^* = \vec{w}_j - I_j^{-1}(\vec{r}_j \times (x\hat{n})). \tag{A.10}$$

From the coefficient of restitution, we know that

$$-\frac{(\vec{v}_i^* + \vec{\omega}_i^* \times \vec{r}_i)\cdot\hat{n} - (\vec{v}_j^* + \vec{\omega}_j^* \times \vec{r}_j)\cdot\hat{n}}{v_i - v_j} = 1. \tag{A.11}$$

Equations A.7 until A.11 give us a system of five equations in five unknowns $\vec{v}_i^*, \vec{v}_j^*, \vec{\omega}_i^*, \vec{\omega}_j^*, x$ which we can solve for x. By inserting equations A.7 until A.10 into equation A.11 and simplifying, we get

$$(v_i - v_j) + \left(\frac{1}{M_i} + \frac{1}{M_j} + \Big((I_i^{-1}(\vec{r}_i \times \hat{n})) \times \vec{r}_i + (I_j^{-1}(\vec{r}_j \times \hat{n})) \times \vec{r}_j\Big)\cdot\hat{n}\right) x = -(v_i - v_j). \tag{A.12}$$

This means that

$$x = \frac{-2(v_i - v_j)}{\frac{1}{M_i} + \frac{1}{M_j} + ((I_i^{-1}(\vec{r}_i \times \hat{n})) \times \vec{r}_i + (I_j^{-1}(\vec{r}_j \times \hat{n})) \times \vec{r}_j)\cdot\hat{n}}. \tag{A.13}$$

Spherical moment of inertia

Let us assume that the particles we are studying have a spherical moment of inertia. Then the matrices I_i, I_j are diagonal with equality between all diagonal entries. Define $C_i = I_i(1,1)$ and $C_j = I_j(1,1)$.

It can easily be shown that for all vectors $\vec{a}$ and $\vec{b}$, the identity $((\vec{a} \times \vec{b}) \times \vec{a}) \cdot \vec{b} = \|a \times b\|^2$ holds. Hence, in this case,

$$x = \frac{-2(v_i - v_j)}{\frac{1}{M_i} + \frac{1}{M_j} + C_i^{-1}\|\vec{r}_i \times \hat{n}\|^2 + C_j^{-1}\|\vec{r}_j \times \hat{n}\|^2}. \tag{A.14}$$

This finishes the derivation of the collision formulas.

Appendix B

Parallel algorithm

Algorithm 1 Main algorithm

```
Initialize the system.
Possibly read input.
Initialize the parallel structures.
Divide the system over the processors.
Initialize the real collisions.

While the maximal time has not been reached do
    Find local event time.
    Communicate the first event time to all other processors.
    Find the global next event time.
    If this is on the current processor do
        Change the characteristics of the involved particles.
        Communicate the changed particles to neighbouring processors.
        Finish handling the event.
    Else do
        Possibly receive particle messages.
        Process the received particle information:
            Update cell lists.
            Update oriented bounding boxes.
            Update events.
    End if
End while

Communicate results.
If this is processor zero do
    Output results.
End if
```

Algorithm 2 Cell crossing

```
If a particle crossed to another cell do
    If the particle entered a virtual cell do
        Remove events with virtual particles.
    Else do
        Possibly add particle collisions.
        Calculate the cell boundary crossing time.
    End if
End if
```

Appendix C

Program documentation

In order to run a simulation, the following characteristics of the system must be given as input.

- Particle shape. Already included shapes are spheres, tetrahedra and twisted triangular prisms. For other polyhedral shapes, an array has to be filled in describing the endpoints of the triangles on the shape relative to the center of mass.
- Number of particles. Up to 4,000 particles works on any system. For more particles, depending on the system, it can be necessary to make some global variables local in order to avoid memory problems.
- Length of the simulation box in the x-, the y- and the z-direction.
- Required number of collisions or amount of time inside the simulation.
- Whether or not to use nearest neighbour lists.
- Whether or not to use twisted periodic boundary conditions, and if so, in which direction.
- How often the program should print a saved state.

The additional information that is printed in a saved state is the following.

- Current time in the simulation.
- Current number of collisions.
- Whether or not the measurements have begun, and if so, at what time.
- Current pressure and torque.
- Additional information such as the number of events and the average number of iteration steps for Conservative Advancement.

Given the name of the file, the program can read the starting configuration of the particles from a file. It can also read all characteristics described above from a saved state file and continue the simulation from there.