Master Thesis

Phase Behaviour of Conical Colloids.

Joep van den Hoven
Universiteit Utrecht.

Supervisors
Marjolein Dijkstra
Simone Dussi
Nick Tasios

Examiners
René van Roij
Rob Bisseling

July 11, 2016

Contents

1 Introduction **3**

2 Monte Carlo Methods **3**

2.1 Metropolis-Hastings Algorithm 4

2.2 NPT ensemble 5

2.3 Order Parameters 7

3 Space Partitioning **8**

3.1 Introduction 9

3.2 Cell List 9

3.3 AABB 9

3.4 Augmented k-d Tree 10

3.5 Near-Neighbour List 15

4 Shapes & Overlap Detection **15**

4.1 GJK 16

4.2 Cone 18

4.3 Ice Cream Cones 20

4.4 Infinite Cones 25

4.5 Double Cones 34

5 Results **36**

5.1 Single Cones 37

5.2 Ice Cream Cones 46

5.3 Double Cones 51

5.4 Double Ice Cream Cones 58

6 Conclusion **60**

6.1 Outlook 63

1 Introduction

In this thesis we investigate the phase behaviour of conical colloidal particles and colloids of several related shapes. The goal is to gain a better understanding of the effects of shape on the liquid crystal phase behaviour of colloidal particles.

We focus on the liquid crystal phases, which are thermodynamic phases in between the fully disordered isotropic phase and the fully ordered crystal phase. Onsager's theoretical study on (infinitely) long hard rods [1] provides a famous example of how a transition between an isotropic phase to a nematic liquid crystal can be driven by particle shape alone. His theoretical predictions were later proven using computer simulations, showing that for sufficiently elongated particles liquid crystal phases are indeed thermodynamically stable.

In particular we investigate which liquid crystal phases are stabilized by conical particles, and how the isotropic-nematic phase transition depends on the particle aspect ratio. In addition, we investigate how the roundedness of the particle affects this behaviour. Finally we also investigate the behaviour of double cones (two cones glued tip to tip).

Cones were chosen because they seemed suitable for computer simulations. They also share many similarities with spherocylinders, which have been studied extensively as a model system for liquid crystals [2], they are both uniaxial, convex, and based on a degenerate quadric (surface defined by a second order polynomial). The main difference is that the cone has a sharp tip, and no up-down symmetry. This asymmetry might lead to the formation of new phases, such as the gyroid phase observed for tapered conical particles [3]. Cone-shaped colloids have also recently been synthesized [4]. Therefore, a more theoretical prediction of their behaviour could be helpful in experiments.

For double cones the mutually excluded volume between two particles depends highly on the orientation of the two cones, which could result in interesting behaviour. For example, it has been suggested [5] that a dumbbell-like shape might be capable of forming a cholesteric phase, this should also apply to double cones.

Monte Carlo simulations of hard particles have been around since 1953 when Metropolis et al. used them to study hard spheres [6]. Since then they have become increasingly powerful, making them the current method of choice for predicting the behaviour of colloidal particles. Like in 1953, however, our calculations will mostly be based on the same Monte Carlo Metropolis-Hastings method, which by now hasn been generalized and applied to a far wider range of problems (e. g. [7, Chapter 3]).

Of course, some new methods have since been discovered to treat interactions between hard particles, such as the GJK algorithm [8] for finding overlap between convex particles, and space partitioning methods such as the cell-list and the near-neighbour-list with which it is possible to quickly find which pairs of particles are close enough to interact [9].

In this thesis we propose a new space-partitioning method for finding interacting particle pairs, by creating a data-structure for quickly finding intersecting boxes. This data-structure was created by combining the augmented binary (interval) tree [10, Section 14.3], which can be used to find intersecting intervals, and the k-d tree which is a generalization of the binary tree for multidimensional data [11]. We also develop novel methods for efficiently detecting overlap between various cone related particle shapes.

2 Monte Carlo Methods

For simulating systems of hard particles, the most convenient method is to use Monte Carlo simulations in the NPT ensemble. The goal is to sample the configuration space, with a distribution corresponding to a system at equilibrium with a reservoir at a fixed temperature, pressure and number of particles. This distribution is proportional to:

$$V^N \exp(-\beta PV - \beta E) \tag{2.1}$$

where N is the number of particles, V is the volume, P is the pressure, E is the (potential) energy and $\beta^{-1} = k_B T$, with k_B the Boltzmann constant and T the temperature. In the NPT ensemble the values of N, P and T are kept fixed. Such a Monte Carlo simulation can then be used to determine how E and V depend on N, P and T. Other ensembles choices such as NVT, in which N, V and T are fixed are also possible. For Monte Carlo simulations it is convenient to use NPT since there are easy ways to measure V and E.

2.1 Metropolis-Hastings Algorithm

To sample the configuration space with distribution (2.1), the simplest, and by far most commonly used, method is the Metropolis-Hastings method. In this method a Markov chain is constructed which has the property that sampling the states of this Markov chain will (eventually) be equivalent to sampling a 'target' distribution. In this Markov chain a new state is chosen randomly according to some 'proposal' distribution (one that is easy to sample), which is then accepted or rejected with a certain probability. If the new state is accepted the system is transitioned to this new state.

If μ is the 'target' distribution and the probability (distribution) of choosing new state B when the system is in state A is given by $g(A \to B)$, then the probability of accepting a move from A to B should be:

$$\operatorname{acc}(A \to B) = \min\left(1, \frac{\mu(B) g(B \to A)}{\mu(A) g(A \to B)}\right). \tag{2.2}$$

This probability is constructed to ensure that:

$$\mu(A) \operatorname{acc}(A \to B) g(A \to B) = \mu(B) \operatorname{acc}(B \to A) g(B \to A) \tag{2.3}$$

which is known as the 'detailed balance condition', and ensures that the rate at which the system jumps from state A to B is the same as the rate at which the system jumps from state B to A. Provided the system is ergodic (every non-trivial region of the configuration space can reach every other non-trivial region) this condition ensures that $\mu(A)$ is the unique stable distribution for the Markov chain we have just constructed. There are other possible choices for the acceptance probability that ensure the same thing, but those typically reject more moves (which can be expensive). This particular choice is referred to as the Metropolis choice.

Ergodic theory and the theory of Markov chains, tell us that as long as the system is ergodic the 'average' distribution of states over time will approach μ. To be more precise, let A_i be the state of this Markov chain at time i then for any function f we have:

$$\lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} f(A_i) = \int f(x) \, \mathrm{d}\mu(x) \tag{2.4}$$

with probability 1, as long as the right side is well defined.

2.2 NPT ensemble

Applying the Metropolis-Hastings algorithm (2.2) to the NPT ensemble distribution (2.1) we find:

$$\text{acc}(A \to B) = \min\left(1, \left(\frac{V_B}{V_A}\right)^N \exp\big(-\beta P(V_B - V_A) - \beta(E_B - E_A)\big)\frac{g(B \to A)}{g(A \to B)}\right) \tag{2.5}$$

where V_A, V_B are the volumes at state A and B respectively, and E_A, E_B are the respective energies. The energy for 'hard' particles depends on the positions of the particles, and is given by:

$$E_A = \begin{cases} \infty & \text{some particles in A overlap} \\ 0 & \text{no particles in A overlap} \end{cases} \tag{2.6}$$

This in turn suggests that (provided A does not contain overlapping particles) $\text{acc}(A \to B)$ is quite simply 0 if B has overlapping particles. Conversely if neither A nor B has overlapping particles and V_A and V_B are equal, then equation (2.2) beaomes:

$$\begin{aligned} \text{acc}(A \to B) &= \min\left(1, \left(\frac{V_B}{V_A}\right)^N \exp\big(-\beta P(V_B - V_A) - \beta(E_B - E_A)\big)\frac{g(B \to A)}{g(A \to B)}\right) \\ &= \min\left(1, \exp\big(-\beta(0-0)\big)\frac{g(B \to A)}{g(A \to B)}\right) \\ &= \min\left(1, \frac{g(B \to A)}{g(A \to B)}\right) \end{aligned} \tag{2.7}$$

so if we ensure that $g(A \to B) = g(B \to A)$ (for instance by moving particles in an isotropic manner) we can accept all moves that do not cause particles to overlap.

Moving to a state with a different volume can be done by choosing the 'new' volume according to a uniformly random distribution on $[V_A - \delta, V_A + \delta]$ where δ is some small constant. This way we ensure that $g(A \to B) = g(B \to A)$ and therefore:

$$\begin{aligned} \text{acc}(A \to B) &= \min\left(1, \left(\frac{V_B}{V_A}\right)^N \exp\big(-\beta P(V_B - V_A) - \beta(E_B - E_A)\big)\frac{g(B \to A)}{g(A \to B)}\right) \\ &= \min\left(1, \left(\frac{V_B}{V_A}\right)^N \exp\big(-\beta P(V_B - V_A)\big)\right) \end{aligned} \tag{2.8}$$

again assuming neither state has overlapping particles.

The volumes V_A and V_B can be made unit-less by dividing them by the particle volume v_p. This results in the following acceptance probability:

$$\text{acc}(A \to B) = \min\left(1, \left(\frac{V_B}{V_A}\right)^N \exp\left(-\beta P v_p \frac{V_B - V_A}{v_p}\right)\right). \tag{2.9}$$

Writing it in this way has the advantage that it no longer matters what length-scale we use. In fact, since the parameters β, P and v_p only appear in the product $\beta P v_p$ we only need to know the value of the unit-less quantity $\beta P v_p$, which is called the reduced pressure. A system of (identical) hard particles is therefore completely determined by the number of particles N, the shape of the particles, and the reduced pressure $\beta P v_p$.

Implementation details

In the implementation used in this thesis each Monte Carlo step consists of picking a random particle and translating it, picking a random particle and rotating it, each repeated N times (where N is the number of particles). After that a volume change is attempted. Each of these changes is accepted or rejected according to the probabilities given in the previous section. This procedure is summarized in algorithm 1.

Algorithm 1 Monte Carlo Step

```
procedure MonteCarloStep
    for i ← 1...N do
        x ← random particle
        MoveParticle(x)
        x ← random particle
        RotateParticle(x)
    ChangeVolume
```

The procedure MoveParticle(x) tries to move the particle x by picking a point in a ball of radius r, according to a uniform distribution, and moving x by this amount. It then checks if this particle overlaps with any of the other particles, and rejects the move if it does. The radius r is tuned to ensure that 30% of the moves are accepted.

Similarly the procedure RotateParticle(x) rotates the main axis of the particle by picking a point on the unit sphere (again uniformly distributed) such that the the angle between the two axes is at most some small angle θ, and changing the axis to this new axis. The procedure also checks for overlap and accepts the move only if it causes no particles to overlap. The axis θ is again tuned to ensure that 30% of the moves are accepted.

As stated, the procedure ChangeVolume changes the volume by sampling a new volume from a uniform distribution on $[V - \delta, V + \delta]$ where δ is some small constant (again tuned to ensure that 20% of the volume changes are accepted). Here we only consider rectangular systems with periodic boundary conditions. Changing the volume is achieved by rescaling the particle positions and the positions of the (periodic) boundaries.

To avoid situations where the pressure is anisotropic, the size of the system is sometimes changed in only one direction (chosen at random), this is done 50% of the time, the other 50% of the time the size of the system is changed in all three directions simultaneously.

Tuning the parameters is done by raising / lowering the value of the parameters by 5% each step, for the first half of the simulation, depending on whether the acceptance probability is too low / high. Measurements done during this first half of the simulation are discarded.

2.3 Order Parameters

To distinguish between different phases it is convenient to use an order parameter. These are typically 0 in the disordered phase and 1 in the ordered phase. They are defined using an ensemble average (denoted by $\langle\cdot\rangle$), which for the NPT ensemble is the expected value for a random configuration with the distribution defined by (2.1). This can be approximated by sampling this distribution using the techniques outlined in the previous section.

A good order parameter for detecting the isotropic-nematic phase transition is:

$$S = \left\langle \frac{3}{2}\cos(\theta)^2 - \frac{1}{2} \right\rangle \tag{2.10}$$

where θ is the angle between the (main) axis of a particle with the nematic director (i. e. the orientation which maximizes the above quantity). When the orientations of the particles are distributed uniformly on the unit sphere, then $\cos(\theta)$ is uniformly distributed with values in $[-1, 1]$, hence the average of $\cos(\theta)^2$ is $\frac{1}{3}$ and S is equal to 0. Conversely, when all particles are perfectly aligned with the nematic director, then $\cos(\theta)$ is 1, hence S is 1 as well. Therefore, S is 0 in the isotropic phase and jumps to some non-zero value in the nematic phase, and gradually gets closer to 1 as the particles become more aligned.

To find the nematic director, $\vec{d}$, it is convenient to rewrite the above expression as follows:

$$\begin{aligned} S &= \left\langle \frac{3}{2}\cos(\theta)^2 - \frac{1}{2} \right\rangle \\ &= \left\langle \frac{3}{2}\left(\vec{d}\cdot\vec{n_i}\right)^2 - \frac{1}{2} \right\rangle \\ &= \left\langle \vec{d^T}\left(\frac{3}{2}\vec{n_i}\vec{n_i}^T - \frac{1}{2}\mathbb{I}\right)\vec{d} \right\rangle \\ &= \vec{d^T}\left\langle \frac{3}{2}\vec{n_i}\vec{n_i}^T - \frac{1}{2}\mathbb{I} \right\rangle\vec{d} \end{aligned} \tag{2.11}$$

where $\vec{n_i}$ is the orientation (i. e. a unit vector pointing along the long axis) of the particle. Writing

$$Q = \left\langle \frac{3}{2}\vec{n_i}\vec{n_i}^T - \frac{1}{2}\mathbb{I} \right\rangle, \tag{2.12}$$

we see that d should be the value maximizing $\vec{d^T}Q\vec{d}$, and since it is an orientation we should also require $\vec{d}^2 = 1$. Using Lagrange multipliers this suggest that $\vec{d}$ satisfies the equation

$$\nabla\left(\vec{d^T}Q\vec{d}\right) = 2\vec{d}Q = \lambda\vec{d}, \tag{2.13}$$

for some $\lambda \in \mathbb{R}$. In other words $\vec{d}$ is an eigenvector of Q, in fact $\vec{d^T}Q\vec{d} = \vec{d^T}\lambda\vec{d} = \lambda$ so it is the eigenvector corresponding to the *largest* eigenvalue and S is the largest eigenvalue. This can be used to find the value of S directly without calculating $\vec{d}$.

Detecting configurational periodicity in the system can be done by looking at the Fourier transform of the particle positions. Denoting the particle position by $\vec{r_i}$ this quantity can be written:

$$\frac{1}{N}\sum_{i=1}^{N} e^{i\vec{k}\cdot\vec{r_i}} \tag{2.14}$$

however, written like this the value depends on the absolute position of the particles, since the system is translationally invariant the average is always 0. This can be fixed by considering only the relative positions:

$$\left\langle \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} e^{i\vec{k}\cdot(\vec{r}_i - \vec{r}_j)} \right\rangle = \left\langle \frac{1}{N^2} \sum_{i=1}^{N} e^{i\vec{k}\cdot\vec{r}_i} \sum_{j=1}^{N} e^{-i\vec{k}\cdot\vec{r}_j} \right\rangle = \left\langle \left| \frac{1}{N} \sum_{i=1}^{N} e^{i\vec{k}\cdot\vec{r}_i} \right|^2 \right\rangle \tag{2.15}$$

which suggests that we should average the squared norm, instead of the actual Fourier coefficient.

If the particle positions are distributed uniformly, then this value is 0. If particles form layers orthogonal to $\vec{k}$ with a distance $2\pi/|\vec{k}|$ between the layers then it is 1. If the orientation and distance of the layers are not known then they can be found by finding the $\vec{k}$ which maximizes the order parameter.

Taking into account rectangular periodic boundary conditions with a box of dimensions (b_1, b_2, b_3) can be done by ensuring that $2\pi k_i b_i \in \mathbb{Z}$. In that case $e^{i\vec{k}\cdot\vec{r}}$ will be the same for each possible image of the particle. Otherwise averaging $e^{i\vec{k}\cdot\vec{r}}$ over enough images of the same particle would make the value arbitrarily close to 0.

Therefore we use the following quantity to detect periodicity:

$$\zeta_i = \left\langle \max_{d_j} \left| \frac{1}{N} \sum_{i=1}^{N} e^{2\pi i r_{ij}/d_j} \right|^2 \right\rangle \tag{2.16}$$

with $d_i \in \{b_i/n_i : n_i \in \mathbb{Z}\}$. In the smectic phase we expect to see periodicity in just one direction, with a d_i slightly bigger than the length of the particle, and for a crystal we expect periodicity in all three.

3 Space Partitioning

3.1 Introduction

To get good performance it is necessary to quickly detect whether particles overlap or not. Especially in large systems this becomes important since with N particles, moving one particle requires $N-1$ overlap checks and changing the volume requires $N(N-1)/2$ overlap checks. However it is not always necessary to check all particle pairs individually. By cleverly 'partitioning' the simulation space into several regions it is possible to quickly rule out parts of the simulation space that are irrelevant.

3.2 Cell List

A popular space partitioning method is the so called 'Cell List'. It works best for spheres, but can be used for other particles by using their bounding sphere.

In this space partitioning methods the space is subdivided into a grid of equally sized cubic cells, where the size of each cell is equal to the diameter of the spheres (it is possible to handle the case where these are not equal, but the basic idea stays the same). We also keep track of which spheres have their centre in a particular cell. This way spheres which overlap must be in neighbouring cells. Hence, to find all spheres overlapping with a certain sphere we only need to check these neighbouring cells, rather than the whole space.

A cell list can be used for some other shapes by calculating bounding spheres (preferably as small as possible) for this shape. A cell list used with these bounding spheres can then be used to restrict the overlap checks to just the particles whose bounding spheres intersect.

One problem with this approach is that for very elongated shapes the volume of the bounding sphere is a lot bigger than that of the shape, and since the cell list can only detect if these bounding spheres may or may not overlap, there will be a lot of cases left to check (although still fewer than without a cell list).

3.3 AABB

For elongated shapes it can be better to use so called 'axis-aligned bounding-boxes' (AABB). These are volumes of the form $[a_1, b_1] \times [a_2, b_2] \times ... \times [a_b, b_n]$. Since these can have different sizes along each axis they can fit elongated particles a lot better (especially if those particles happen to point along a particular axis, which luckily happens quite often).

The so called oriented bounding boxes (OBB) which are rotated versions of the AABBs can fit even better but they are hard to deal with in data-structure; owing to the fact that their axes do not all point in the same direction.

An AABB of the form $[a_1, b_1] \times [a_2, b_2] \times ... \times [a_b, b_n]$ can be stored as a pairs of vectors $\vec{a} = (a_1, \ldots, a_n)$, $\vec{b} = (b_1, \ldots, b_n)$ (corresponding to two opposing corners of the box). Two AABBs $(\vec{a}, \vec{b})$ and $(\vec{c}, \vec{d})$ then intersect if:

$$\forall i : a_i \leq d_i \wedge c_i \leq b_i \tag{3.1}$$

However in some cases it can be more convenient to work with the 'midpoint' and 'size' of

the boxes, defined as follows:

$$\begin{aligned} \vec{m} &= \tfrac{1}{2}(\vec{a} + \vec{b}) \\ \vec{s} &= \tfrac{1}{2}(\vec{b} - \vec{a}). \end{aligned} \tag{3.2}$$

Two boxes with midpoints $\vec{m}_A, \vec{m}_B$ and sizes $\vec{s}_A$ and $\vec{s}_B$ intersect if and only if:

$$\forall i : |\vec{m}_A - \vec{m}_B|_i \leq (\vec{s}_A + \vec{s}_B)_i. \tag{3.3}$$

3.4 Augmented k-d Tree

Calculating the bounding boxes is of course not particularly useful if there is no fast way of finding ones that intersect. This can be done by combining two data structures. The data structures needed are an augmented binary tree used for finding intersecting intervals (i.e. 1-dimensional boxes), combined with the k-d tree which generalizes binary trees for multi dimensional data.

The augmented k-d tree inherits most of its characteristics from the augmented binary tree, which it is based on. Most of the algorithms involving the augmented binary tree can be adapted quite easily for use with a k-d tree instead. In fact it is probably possible to base the augmented k-d tree on different varieties of the augmented binary tree. The version described here is designed to be easy to update, provided the intervals only change by small amounts (which is usually the case in NPT Monte Carlo simulations).

Augmented Binary Tree

The augmented binary tree is essentially a binary tree of all the midpoints of the intervals, 'augmented' with the 'size' of the corresponding interval.

To be specific, this binary tree consists of nodes which each store a midpoint of an interval, and the size of this interval. We will denote the midpoint and size of node i as m_i and s_i, respectively. Each node also has a (possibly empty) left and right sub-tree. The tree will be constructed such that the left sub-tree of a node r only contains nodes i for which $m_i \leq m_r$, and the right sub-tree only contains nodes i for which $m_i \geq m_r$. This requirement can be relaxed slightly, but we will come back to that later.

Since we are now dealing with intervals and not points each node will also need to store some additional information on its sub-trees to ensure no intervals are missed. For each node we need what we shall call the 'margin' of that particular node. For a node r with child nodes T_r this value is:

$$\text{margin}_r = \max_{i \in T_r} s_i - |m_r - m_i|. \tag{3.4}$$

This value is defined such that the intervals in the sub-trees of r might extend beyond the midpoint of r, but not more than a distance of margin_r. Graphically this looks something like figure 1.

After this augmented binary tree has been constructed it is relatively simple to find all intervals intersecting with a particular 'query' interval. At each node we simply check if the distance between the midpoint of the query interval and the 'node' interval is less than the sum of the size of the query interval and the 'margin' of the corresponding sub-tree. If this distance is bigger then by construction the query interval can only intersect with intervals on one side of the current node, meaning we do not need to check one of the sub-trees

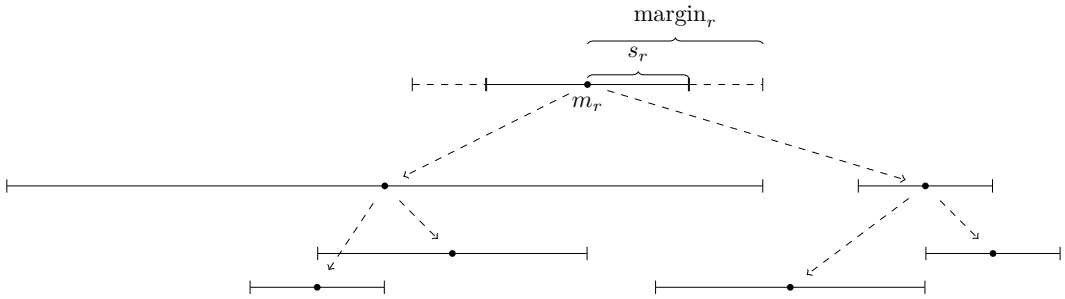


Figure 1: Graphical representation of augmented binary tree.

and the interval at the current node. Otherwise we simply check everything. This leads to algorithm 2.

Algorithm 2 Find intersecting entries

input Interval $[a, b] = [m - s, m + s]$ and node i.
output yields all intervals in sub-tree of node i intersecting with $[a, b]$
procedure FINDOVERLAP(i, $[a, b]$)
  $R \leftarrow m_i - n$
  $S \leftarrow \text{margin}_i + s$
  if $|R| \leq S$ **then**
    FINDOVERLAP(leftTree$_i$, $[a, b]$)
    FINDOVERLAP(rightTree$_i$, $[a, b]$)
    if INTERSECTS(node, $[a, b]$) **then**
      yield node
  else if $R > 0$ **then**
    FINDOVERLAP(leftTree$_i$, $[a, b]$)
  else
    FINDOVERLAP(rightTree$_i$, $[a, b]$)

This algorithm calls the function INTERSECTS, which uses equation (3.3) to check if the interval of a node intersects with the query interval. It also refers to leftTree$_i$ and rightTree$_i$ which are the roots of the left and right sub-tree of i, respectively. If these do not exist the corresponding call to FINDOVERLAP is skipped.

With this we now have the most important part of the data structure. All that is left is how to build and maintain the data structure. To avoid being repetitive we skip this part for the binary tree and go directly to the k-d tree.

k-d Tree

Since there is no huge difference between building the k-d tree and the augmented k-d tree (apart from the additional information added to each node) this is a good opportunity to describe what a k-d tree is and how to build one.

A k-d tree is nothing else but a binary tree for storing multi-dimensional points where, instead of sorting the nodes along the same dimension, the nodes are sorted along a different dimension for each level. For example the top level will be sorted along dimension 1, while

the next will be sorted along dimension 2 etc. This ensures that points that are close to one another are 'close' in the data structure as well.

For the performance of operations on the k-d tree it is important that it is balanced. The easiest way to ensure this is to construct it using the median-cut algorithm, which partitions the tree exactly along the median, ensuring that the tree has an (almost) equal number of nodes at each side. Algorithm 3 describes how to do this when the nodes are laid out in a list.

Algorithm 3 Median-Cut

```
procedure InitializeTree
    MedianCut(0, N − 1, 0)
procedure MedianCut(l, r, d)
    if l ≤ r then
        C ← ⌊(l+r)/2⌋
        QuickSelect(l, r, C, d)
        MedianCut(l, C − 1, d + 1 (mod n))
        MedianCut(C + 1, r, d + 1 (mod n))
procedure QuickSelect(l, r, c, d)
    P ← Partition(l, r, d)
    L ← l
    R ← r
    while P ≠ c do
        if P > c then
            R ← P − 1
        else
            L ← P + 1
        P ← Partition(L, R, d)
```

The function MedianCut takes a part of the list of intervals (represented as a list of midpoints $(m_0, m_1, ..., m_{n-1})$ and sizes $(s_0, s_1, ..., s_{n-1})$) starting at index l and ending at index r, and reorders them such that for everything left of the middle index $C = \lfloor \frac{l+r}{2} \rfloor$, i. e. for all $i < C$:

$$m_{id} \leq m_{Cd} \tag{3.5}$$

and for all $i > C$:

$$m_{id} \geq m_{Cd} \tag{3.6}$$

To achieve this it uses the function Partition(l, r, d), which reorders the list starting from index l up to (and including) index r such that:

$$\forall i < P : m_{id} \leq m_{Pd} \tag{3.7}$$

$$\forall i > P : m_{id} \geq m_{Pd} \tag{3.8}$$

for *some* index $l \leq P \leq r$ (which it returns as output), which might not be equal to C. Such a function is generally known as a partition function. For performance reasons the particular partition function used in this project uses a slightly more complicated algorithm than other implementations of the partition function. The one used in this project, and optimized for this particular purpose is described in algorithm 5 on page 15.

After running MedianCut() on the whole list, the list is implicitly a tree, where each sub-tree is represented by range of indices $l, l + 1, .., r$ with the root at $\lfloor \frac{l+r}{2} \rfloor$. This concludes

the building of the k-d tree. Which is the last part necessary for describing the augmented k-d tree.

Augmented k-d Tree

Building the augmented k-d tree is not fundamentally different from building the k-d tree, the main difference is that we also need to calculate the 'margin' for each sub-tree, which can be done as part of the Median-cut algorithm.

Finding intersecting boxes is also not that much different from finding intersecting intervals. Algorithm 2 can be used with almost no change. Even though we are only looking along a single dimension each time we can still justify skipping the same nodes, because for two boxes to intersect the corresponding intervals should intersect in every single dimension.

The only remaining part is maintaining the data-structure. A simple method would be to rebuild the whole tree every time we change something, but that is rather expensive. It is better to check what part of the data-structure is invalidated by the change and only rebuild that particular part. In particular we want to ensure that the updated node is on the correct side of each parent node, and all child nodes are still on the correct side of the updated node. We also want to make sure that the value of 'margin' is still valid. In the following we assume that MedianCut takes care of calculating the margin. The procedure for updating a particular entry is outlined in algorithm 4.

Algorithm 4 Update Entry

```
procedure Update(i, m, s)
    m_i ← m
    s_i ← s
    margin_i ← max(margin_i, s_i)
    Rebuild(0, N − 1, 0, i)
procedure Rebuild(l, r, d, i)
    C ← ⌊(l+r)/2⌋
    S ← margin_C − s_id
    if i < C and m_id < m_Cd + S then
        Rebuild(l, C − 1, d + 1 (mod n), i)
    else if i > C and m_id > m_Cd − S then
        Rebuild(C + 1, r, d + 1 (mod n), i)
    else if i ≠ C then
        MedianCut(l, r, d)
    else if not CheckMedian(l, r, d) then
        MedianCut(l, r, d)
function CheckMedian(l, r, d)
    C ← ⌊(l+r)/2⌋
    for l ≤ i < C do
        if m_id > m_Cd + (margin_C − s_id) then
            return false
    for C < i ≤ r do
        if m_id < m_Cd − (margin_C − s_id) then
            return false
    return true
```

This algorithm takes as input the position in the list of the interval to the updated i, its new midpoint m and its new size s. The procedure REBUILD then goes through the tree and checks if the new interval is still on the correct side of each parent interval. Note that it takes into account the value of 'margin' allowing the midpoint of the new interval to be slightly on the 'wrong' side, as long as the interval does not extend more than a distance of 'margin' beyond the parent interval. This does not affect the result of FINDOVERLAP. If at any point the interval is too far on the 'wrong' side, that particular part of the tree is rebuilt entirely.

If the updated node is on the right side of all parent nodes then the procedure CHECKMEDIAN is called to check if all child nodes are on the correct side of the updated node. Again the margin is taken into account to avoid unnecessary rebuilding of the tree. CHECKMEDIAN allows the midpoint of a particular node to be on the wrong side, as long as the margin is big enough that algorithm 2 still works.

Implementation Notes

- As mentioned REBUILD and CHECKMEDIAN take into account the margin to avoid unnecessary rebuilds, this can be exploited by making margin slightly larger (say 5%) than it needs to be. This causes the tree to be rebuild even less often. By doing this the time spent rebuilding the tree becomes negligible, which saves a lot of time, especially in big systems.
- Instead of using a symmetric 'margin' it is also possible to use a separate one for the left and right sub-trees, but since in practice the value of 'margin' is nearly always the same as the size of the corresponding node the benefits are minimal and entirely negated by the increased complexity of algorithm 2.
- For the best performance PARTITION should be implemented in a way that leaves the order of the list intact as much as possible, since most of the items are still be in the correct position. Unfortunately most implementations of the partition function either reorder most of the list, or do not keep track of the position of the 'pivot'. A stable variant of the partition function is given in algorithm 5.
- Dealing with periodic boundary conditions can be done by checking all images of the query algorithm that could possibly intersect with any of the boxes. To do this it is also necessary to keep track of the maximum distance each box extends outside the boundary.
- When moving particles with periodic boundary conditions the particle should be moved to it is closest possible image. Large jumps in position should be avoided because they would invalidate large parts of the k-d tree (this problem could be alleviated by using a more sophisticated algorithm, but if at all possible it is easier to avoid the issue entirely).

Algorithm 5 Stable Partition Function

```
function PARTITION(l, r, d)
    c ← ⌊(l+r)/2⌋
    p ← m_cd
    while l < r and c < r do
        while m_ld < p do
            l ← l + 1
        while m_rd > p do
            r ← r − 1
        SWAP(l, r)
    if l < c then
        c ← r
        for i = c downto l do
            if m_id > p then
                SWAP(i, c)
                c ← c − 1
        SWAP(l, c)
    else if r > c then
        c ← l
        for i = c to r do
            if m_id < p then
                SWAP(i, c)
                c ← c + 1
        SWAP(r, c)
    return c
```

3.5 Near-Neighbour List

In some cases it is possible to speed up things even more by storing which particles had intersecting bounding boxes, in a so called 'near neighbour list'. Although this is only useful if the 'new' bounding box is inside the old one, which is pretty rare. However by using a bounding box which is slightly too big this chance can be increased significantly. The downside is that this will require more overlap checks, but it has the advantage that FINDOVERLAP will not have to be called as often. Unfortunately (or fortunately, depending on your point of view) in most cases the method FINDOVERLAP is sufficiently fast that the savings tend to be negligible unless there is a really fast method to detect overlap between individual particles.

4 Shapes & Overlap Detection

4.1 GJK

The most general method used in this thesis for detecting overlap is GJK. GJK was invented by Gilbert, Johnson and Keerthi in 1988 [8], and is an algorithm for detecting whether a convex set contains the origin.

GJK can also be used to detect whether two convex sets intersect, which is more relevant for our case. This can be done using the Minkowski difference, which for two sets A and B is defined as follows:

$$A - B := \{\vec{a} - \vec{b} : \vec{a} \in A, \vec{b} \in B\} \tag{4.1}$$

by definition this set only contains the origin if the intersection of A and B is not empty. It is also important to note that as long as A and B are both convex then $A - B$ will be as well. Hence if we use the GJK algorithm on $A - B$ it tells us whether A and B intersect.

Before moving on to the algorithm we should first introduce some terminology.

Terminology

convex set A convex set is a set $X \subset \mathbb{R}^n$ with the property that if $\vec{x} \in X$ and $\vec{y} \in X$ then $(1-t)\vec{x} + \vec{y} \in X$ for all $t \in [0, 1]$.

support function The support function of a convex set X is defined as follows:

$$\vec{s}_X(\vec{d}) = \underset{\vec{x} \in X}{\operatorname{argmax}}\, \vec{x} \cdot \vec{d} \tag{4.2}$$

In case the maximum is not unique then it does not matter which one is chosen. Also note that if s_A and s_B are the support functions of sets A and B respectively, then

$$s_{A-B}(\vec{t}) = s_A(\vec{t}) - s_B(\vec{t}) \tag{4.3}$$

is the support function of $A - B$.

convex combination A convex combination of a collection of points $\vec{x}_1, \ldots, \vec{x}_k$ is a linear combination $\sum_{i=1}^{k} w_i \vec{x}_i$ with the property that $\sum_{i=1}^{k} w_i = 1$ and all $w_i \geq 0$.

convex hull The convex hull of a set of points X is the set of all convex combinations of X. It is also the smallest convex set containing X.

The Algorithm

To find whether a convex set X contains the origin the GJK algorithm goes through the following steps:

- Guess point $\vec{d}$ in X close to origin.
- If $\vec{d} = 0$ then X contains the origin.
- Calculate $\vec{x} = s_X(-\vec{d})$.
- If $\vec{x} \cdot \vec{d} > 0$ then X does not contain origin
- Else, use x to improve guess.

Since $s_X(-\vec{d}) \cdot \vec{d}$ is be definition minimal it is clear that $\vec{x} \cdot \vec{d} > 0$ implies that the convex set contains no points $\vec{x}$ with $\vec{x} \cdot \vec{d} = 0$, in particular it does not contain the origin.

Ideally the point $\vec{d}$ would be the point in X closest to the origin, in that case the algorithm would always terminate in a single step, however it is hard to find this point directly. Hence, X is approximated by a convex hull of points instead. Since we are only interested in the part of the convex hull closest to the origin we can discard points from this set as long as this does not increase the distance from the convex hull to the origin. This keeps the number of points in the set at or below the number of dimensions.

It turns out that if we include the point $\vec{x} = s_X(-\vec{d})$ (where $\vec{d}$ is the point in the convex hull closest to the origin) in the convex hull then the distance from the convex hull to the origin always decreases (unless $\vec{d}$ is the point *in* X closest to the origin). This results in algorithm 6.

Algorithm 6 GJK

```
function GJK(s)
    d ← arbitrary direction
    S ← {}
    while d² > 0 do
        x ← s(−d)
        if x · d > 0 then
            return false
        S ← S ∪ {x}
        d ← Closest(S)
        S ← Minimize(S)
```

Algorithm 6 needs two additional functions, the function Closest(S) which calculates the point on the convex hull of S closest to 0 and the function Minimize(S) which returns the smallest subset of S for which the convex hull contains this point. In practice it can be convenient to combine these two steps. For a more detailed description of this process, and some comments on its numerical stability, please refer to [12].

4.2 Cone

The simplest shape used in this thesis is the single cone with a flat base. Such a cone can be considered the surface of revolution of a simple triangle, which is shown in figure 2. In this figure the length L and diameter D of the cone are defined, which are used to define the aspect ratio L/D. The aspect ratio together with the particle volume are enough to define the shape entirely. By defining volume and pressure in terms of the particle volume the particle volume is effectively set to 1.

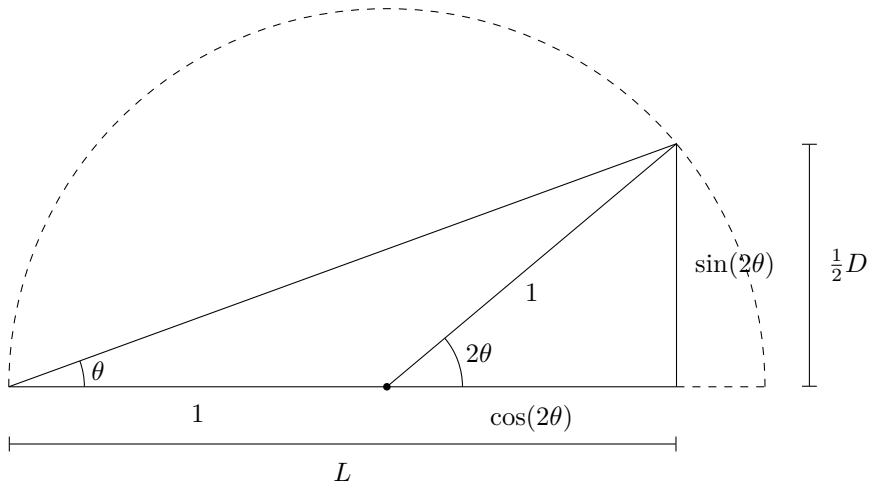


Figure 2: Construction of a single cone.

The construction in figure 2 uses the angle θ. This angle depends directly on the aspect ratio L/D, as follows:

$$\theta = \arctan\left(\frac{D}{2L}\right). \tag{4.4}$$

Volume

A rather simple integration shows that the the volume of a single cone is:

$$v_p = \int_0^L \pi\left(\frac{z}{L}\frac{D}{2}\right)^2 \mathrm{d}z = \frac{\pi}{3}\left(\frac{D}{2L}\right)^2 L^3 = \frac{\pi}{3}L\left(\frac{D}{2}\right)^2 \tag{4.5}$$

Close Packing

As initial configuration for the simulations, it is convenient to start with a dense configuration and then expand. This prevents the system from getting stuck in a meta-stable state. The configuration we will use is based on a lattice, described in [13], generated by the following vectors:

$$\begin{pmatrix} \frac{1}{2}D \\ 0 \\ L \end{pmatrix}, \begin{pmatrix} 0 \\ \frac{2}{3}D \\ -\frac{2}{3}L \end{pmatrix}, \begin{pmatrix} 0 \\ 0 \\ 2L \end{pmatrix}. \tag{4.6}$$

At each lattice points two cones are placed, each with their tip on the lattice point, one pointing up the other pointing down. Placed on such a lattice, the cones achieve a volume fraction of $\pi/4 \approx 0.79$, regardless of the aspect ratio. Figure 3 shows what this configuration looks like for $L/D = 3$.

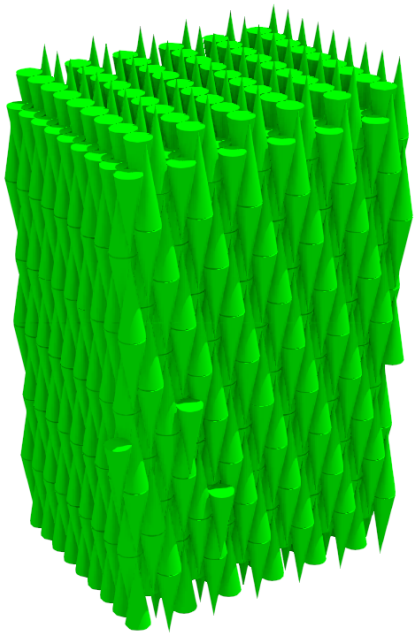


Figure 3: Initial configuration for $L/D = 3$.

Overlap Detection

To detect overlaps between cones we use the GJK algorithm. For this the support function is needed, which for a cone with orientation $\vec{n}$ and position $\vec{p}$, is given by:

$$s(\vec{d}) = \begin{cases} \vec{p} & \frac{\vec{d}\cdot\vec{n}}{|\vec{d}|} < -\frac{D/2}{\sqrt{\left(\frac{D}{2}\right)^2+L^2}} \\ \vec{p} + L\vec{n} + \frac{D}{2}\frac{\vec{d}-\vec{n}(\vec{d}\cdot\vec{n})}{|\vec{d}-\vec{n}(\vec{d}\cdot\vec{n})|} & \frac{\vec{d}\cdot\vec{n}}{|\vec{d}|} \geq -\frac{D/2}{\sqrt{\left(\frac{D}{2}\right)^2+L^2}} \\ \vec{p} + L\vec{n} & \frac{\vec{d}\cdot n}{|\vec{d}|} = 1 \end{cases} \tag{4.7}$$

Checking if the tip of one cone happens to be inside the other can be used to quickly detect overlap in some cases, avoiding the need for the relatively expensive GJK. A point $\vec{q}$ is in a cone with orientation $\vec{n}$ and position $\vec{p}$ if

$$\frac{\left(\vec{q} - \vec{n}(\vec{n}\cdot\vec{q})\right)^2}{(D/2)^2} \leq \frac{(\vec{n}\cdot\vec{q})^2}{L^2} \tag{4.8}$$

Calculating this expression is significantly faster than using the GJK algorithm.

In principle it is possible to avoid GJK entirely by detecting overlap fully analytically. One way of doing this is by parametrizing cones as collections of disks (with a linearly moving centre and linearly increasing radius) and detect overlap between those. However this is only worth doing if it can be done more quickly than GJK.

To find out how computationally expensive it is to find two overlapping disks we first need to find a way of finding two overlapping disks. One way would be to find the two disks closest to each other, but unfortunately no (simple) analytical solution for the distance between two disks seems to have been found at this point. However, since we are not interested in the distance and only whether two disks overlap we can, and since both disks are confined to a plane we only need to check along the line where these planes intersect. On this line the disks reduce to 1-dimensional 'balls' (i. e. an interval) where the midpoint is the point closest to the centre of the disk and the 'radius' is given by $\sqrt{r^2 - d^2}$ where d is the distance from the centre of the disk to the line, and r is the radius of the disk. After that it is a simple matter of checking whether the distance between these two 1 dimensional balls is less than their radii.

For two disks with their centres at x_1 and x_2, orientations n_1 and n_2 and radii r_1 and r_2 respectively the distance to the line of intersection is given by:

$$d_i = \frac{|x_1 - x_2| \cdot n_j}{|n_1 \times n_2|} \tag{4.9}$$

for i and j in $\{1, 2\}$ and $i \neq j$. The distance between the two centres projected onto the line of intersection is given by:

$$\frac{|(x_1 - x_2) \cdot (n_1 \times n_2)|}{|n_1 \times n_2|}. \tag{4.10}$$

Writing $x_1 - x_2 = p$ and combining these equations we find that the disks overlap precisely when:

$$\frac{|p \cdot (n_1 \times n_2)|}{|n_1 \times n_2|} < \sqrt{r_1^2 - \frac{(p \cdot n_2)^2}{(n_1 \times n_2)^2}} + \sqrt{r_2^2 - \frac{(p \cdot n_1)^2}{(n_1 \times n_2)^2}} \tag{4.11}$$

or equivalently:

$$|p \cdot (n_1 \times n_2)| - \sqrt{r_1^2 (n_1 \times n_2)^2 - (p \cdot n_2)^2} - \sqrt{r_2^2 (n_1 \times n_2)^2 - (p \cdot n_1)^2} < 0. \tag{4.12}$$

It is possible, in theory, to find which disks minimize the difference in the above expression, but the square roots make this somewhat difficult. Also, it turns out there is a similar shape where it is even easier to detect collisions, which can also be used to speed up collision checking between cones. This shape is the subject of the next section.

4.3 Ice Cream Cones

Another interesting cone-like shape is one formed by a cone with a sphere on top, which will be referred to as an 'ice-cream cone'. Apart from the tip this shape is differentiable. This shape can be used to examine the effect the flat base of the cone has on the behaviour of the system.

The construction in figure 4 uses the parameter r, which depends directly on the aspect ratio L/D. To be precise L is $2 + r$ and D is simply $2r$ so

$$\frac{L}{D} = \frac{2 + r}{2r} = \frac{2}{r} + \frac{1}{2} \tag{4.13}$$

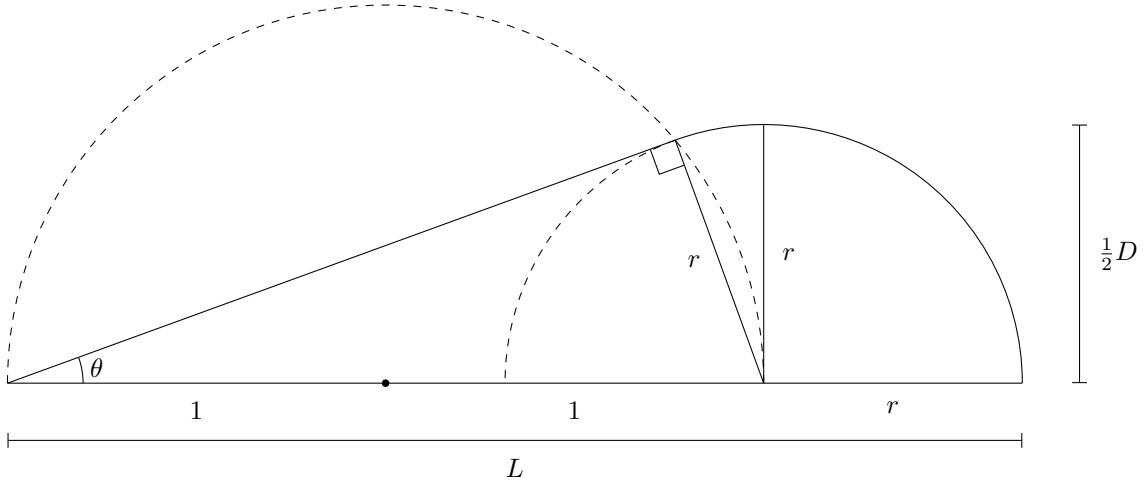


Figure 4: Construction of an 'ice-cream cone'.

hence

$$r = \frac{2}{L/D - \frac{1}{2}}. \tag{4.14}$$

Volume

Calculating the volume of an ice cream cone is somewhat more involved because of the spherical cap. The easiest method is to decompose the shape into a spherical part and a conical part, as in figure 5.

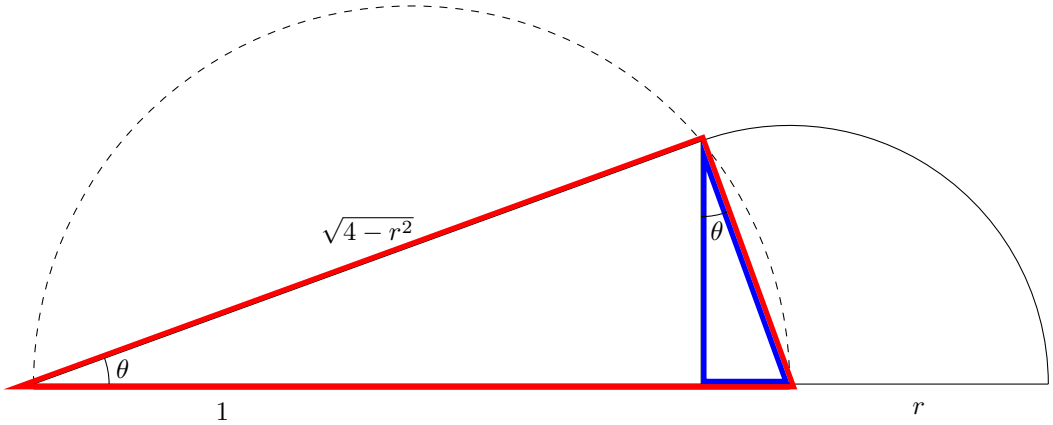


Figure 5: Decomposition into cone with partial sphere.

Finding out the precise height of the cone and the partial sphere is somewhat complicated, note however that the triangles marked red and blue in figure 5 are congruent. Also the long side of the red triangle has length 2, whereas the length of the long side of the blue triangle is r. Therefore the lengths of both triangles are as denoted in figure 6 on the following page.

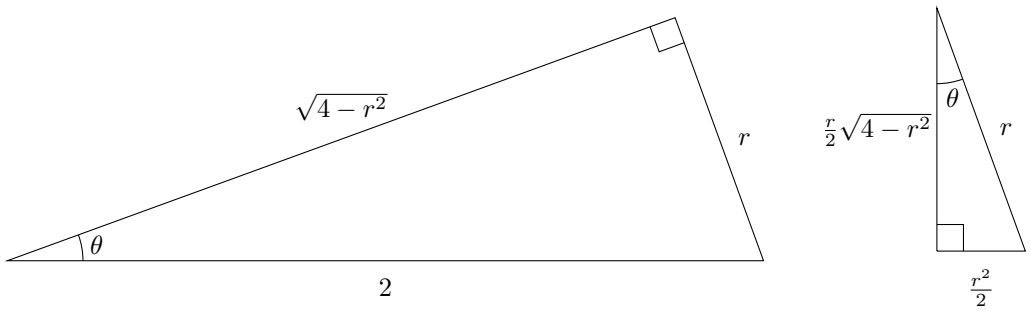


Figure 6: Congruent triangles.

Hence, the cone has length $2 - \frac{r^2}{2}$, and radius $\frac{r}{2}\sqrt{4-r^2}$, and the partial sphere has a total height of $r + \frac{r^2}{2}$ and radius r. This means that the partial sphere has a volume of:

$$\int_{-r}^{\frac{r^2}{2}} \pi(r^2 - z^2)\mathrm{d}z = \left[\pi r^2 z - \frac{\pi}{3}z^3\right]_{-r}^{\frac{r^2}{2}} = \pi r^2\left(\frac{r^2}{2} + r\right) - \frac{\pi}{3}\left(\frac{r^6}{8} + r^3\right). \tag{4.15}$$

A cone with length l and radius r has a volume of $\frac{\pi}{3}lr^2$, hence the cone inside the ice cream cone has a volume of

$$\frac{\pi}{3}\left(2 - \frac{r^2}{2}\right)\frac{r^2}{4}(4 - r^2). \tag{4.16}$$

In total this gives a volume of

$$v_p = \pi r^2\left(\frac{r^2}{2} + r\right) - \frac{\pi}{3}\left(\frac{r^6}{8} + r^3\right) + \frac{\pi}{3}\left(2 - \frac{r^2}{2}\right)\frac{r^2}{4}(4 - r^2). \tag{4.17}$$

Overlap Detection

One advantage of this shape is that there is a relatively simple way to detect overlap. Key to this method is that an ice cream cone is equivalent to the region swept out by sphere moving along a line segment, with a linearly increasing radius (as shown in figure 7). The distance between two spheres is simply given by the distance between their centres minus the sum of their radii. To find out if two ice cream cones overlap it is enough to find the minimum of this distance.

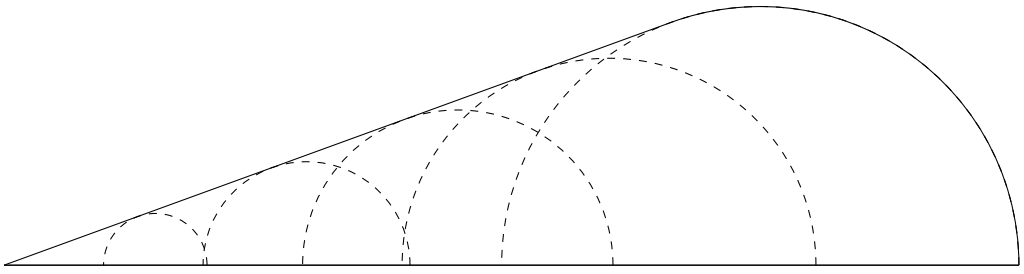


Figure 7: Ice cream cone as a union of spheres.

For a ice cream cone with its tip at $\vec{x}$ and axis $\vec{a}$ (with $a^2 = 1$) the centre of the sphere is at $\vec{x} + t\vec{a}$, and its radius is γt with γ some constant depending on the aspect ratio of the ice cream cone and t some parameter. For the one depicted in figure 4 on page 21 this γ is equal to $r/2$ or $\sin(\theta)$ and t is in the interval $[0, 2]$.

With two ice cream cones with their tips at $\vec{x}$ and $\vec{y}$, axes $\vec{a}$ and $\vec{b}$ and parameters t, s respectively the distance between the centres of the two spheres is:

$$|(\vec{x} - \vec{y}) + t\vec{a} - s\vec{b}| \tag{4.18}$$

and the sum of the radii of the spheres is simply:

$$\gamma(t + s). \tag{4.19}$$

By squaring these and subtracting them we get the following expression:

$$((\vec{x} - \vec{y}) + t\vec{a} - s\vec{b})^2 - \gamma^2(t + s)^2 \tag{4.20}$$

which is a quadratic polynomial in t and s. In some cases it is more convenient to write this as a matrix product. Let $(\vec{x} - \vec{y}) = \vec{p}$, the above equation can the be written:

$$\begin{pmatrix} t & s & 1 \end{pmatrix} \begin{pmatrix} 1 - \gamma^2 & -\vec{a} \cdot \vec{b} - \gamma^2 & \vec{a} \cdot \vec{p} \\ -\vec{a} \cdot \vec{b} - \gamma^2 & 1 - \gamma^2 & -\vec{b} \cdot \vec{p} \\ \vec{a} \cdot \vec{p} & -\vec{b} \cdot \vec{p} & \vec{p}^2 \end{pmatrix} \begin{pmatrix} t \\ s \\ 1 \end{pmatrix} \tag{4.21}$$

To find out if two ice cream cones overlap we need to check if the expression (4.20) becomes negative in the region $(t, s) \in [0, 2] \times [0, 2]$, for this it is enough to just check the minimum of this expression. This minimum will either be a local minimum or occur somewhere on the edge of the region (with either t or s equal to 0 or 2).

To find the local minima we should set the gradient of (4.20) to 0, using the matrix formulation it can be easily shown that this gradient is given by:

$$2 \begin{pmatrix} 1 - \gamma^2 & -\vec{a} \cdot \vec{b} - \gamma^2 & \vec{a} \cdot \vec{p} \\ -\vec{a} \cdot \vec{b} - \gamma^2 & 1 - \gamma^2 & -\vec{b} \cdot \vec{p} \end{pmatrix} \begin{pmatrix} t \\ s \\ 1 \end{pmatrix} \tag{4.22}$$

setting this to 0 is equivalent to:

$$\begin{pmatrix} 1 - \gamma^2 & -\vec{a} \cdot \vec{b} - \gamma^2 \\ -\vec{a} \cdot \vec{b} - \gamma^2 & 1 - \gamma^2 \end{pmatrix} \begin{pmatrix} t \\ s \end{pmatrix} = \begin{pmatrix} -\vec{a} \cdot \vec{p} \\ \vec{b} \cdot \vec{p} \end{pmatrix} \tag{4.23}$$

with the solution:

$$\begin{pmatrix} t \\ s \end{pmatrix} = \frac{1}{(1 - \gamma^2)^2 - (\vec{a} \cdot \vec{b} + \gamma^2)^2} \begin{pmatrix} 1 - \gamma^2 & \vec{a} \cdot \vec{b} + \gamma^2 \\ \vec{a} \cdot \vec{b} + \gamma^2 & 1 - \gamma^2 \end{pmatrix} \begin{pmatrix} -\vec{a} \cdot \vec{p} \\ \vec{b} \cdot \vec{p} \end{pmatrix} \tag{4.24}$$

It should also be noted that this is a minimum if and only if the matrix:

$$\begin{pmatrix} 1 - \gamma^2 & \vec{a} \cdot \vec{b} + \gamma^2 \\ \vec{a} \cdot \vec{b} + \gamma^2 & 1 - \gamma^2 \end{pmatrix} \tag{4.25}$$

is positive definite. Since $1-\gamma^2$ is always positive it is enough to determine whether the determinant $(1-\gamma^2)^2 - (\vec{a}\cdot\vec{b}+\gamma^2)^2$ is positive.

After checking the local minima (if any, and if it lies in the region $(t,s) \in [0,2]\times[0,2]$) the boundary of the region should be checked. Fixing s the equation reduces to a simple quadratic equation. To find the minimum value for $t \in [0,2]$ it is enough to check the value of t closest to the global minimum. Setting the appropriate gradient to 0 this value turns out to be:

$$t = \frac{s(\vec{a}\cdot\vec{b}+\gamma^2) - \vec{a}\cdot\vec{p}}{1-\gamma^2}. \tag{4.26}$$

Similarly if we fix t the minimum is at:

$$s = \frac{t(\vec{a}\cdot\vec{b}+\gamma^2) + \vec{b}\cdot\vec{p}}{1-\gamma^2}. \tag{4.27}$$

Combining all of this results in the algorithm 7.

Algorithm 7 Ice Cream Cone Overlap Detection.

function IccOverlap($\vec{p},\vec{a},\vec{b}$)
$M \leftarrow \begin{pmatrix} 1-\gamma^2 & -\vec{a}\cdot\vec{b}-\gamma^2 & \vec{a}\cdot\vec{p} \\ -\vec{a}\cdot\vec{b}-\gamma^2 & 1-\gamma^2 & -\vec{b}\cdot\vec{p} \\ \vec{a}\cdot\vec{p} & -\vec{b}\cdot\vec{p} & \vec{p}^2 \end{pmatrix}$
$\Delta \leftarrow (1-\gamma^2)^2 - (\vec{a}\cdot\vec{b}+\gamma^2)^2$
$(t,s) \leftarrow \Delta^{-1}\begin{pmatrix} 1-\gamma^2 & \vec{a}\cdot\vec{b}+\gamma^2 \\ \vec{a}\cdot\vec{b}+\gamma^2 & 1-\gamma^2 \end{pmatrix}\begin{pmatrix} -\vec{a}\cdot\vec{p} \\ \vec{b}\cdot\vec{p} \end{pmatrix}$
if $\Delta > 0$ **and** $(t,s) \in [0,2]\times[0,2]$ **and** $(t,s,1)M(t,s,1)^T < 0$ **then**
return true
for all $s \in \{0,2\}$ **do**
$t \leftarrow \frac{s(\vec{a}\cdot\vec{b}+\gamma^2)-\vec{a}\cdot\vec{p}}{1-\gamma^2}$
$t \leftarrow \max(0,\min(2,t))$
if $(t,s,1)M(t,s,1)^T < 0$ **then**
return true
for all $t \in \{0,2\}$ **do**
$s \leftarrow \frac{t(\vec{a}\cdot\vec{b}+\gamma^2)+\vec{b}\cdot\vec{p}}{1-\gamma^2}$
$s \leftarrow \max(0,\min(2,s))$
if $(t,s,1)M(t,s,1)^T < 0$ **then**
return true
return false

Single Cone Overlap (revisited): Since the single cone is contained in the ice cream cone this method can also be used to quickly rule out overlap for cones. Leaving only a small part to be checked with GJK. Although it is in theory possible to do a similar calculation for cones with a flat base (see the previous section) the polynomials involved are of a higher order which makes the calculations rather more complicated. In particular the flat base of the cones makes a fully analytical solution somewhat difficult, since the expression for finding overlap between two disks is more difficult than the expression for two spheres.

Recall that the expression for detecting overlap between two disks was (using the notation from this section):

$$|\vec{p}\cdot(\vec{a}\times\vec{b})| > \sqrt{r_a^2(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2}+\sqrt{r_b^2(\vec{a}\times\vec{b})^2-(\vec{a}\cdot\vec{p})^2}. \tag{4.28}$$

where r_a and r_b are the radii of the disk corresponding to $\vec{a}$ and $\vec{b}$ respectively. To compare this directly to the method for the ice cream cones the square roots should be removed. To do this we first square both sides, this yields:

$$(\vec{p}\cdot(\vec{a}\times\vec{b}))^2 > r_a^2(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2+r_b^2(\vec{a}\times\vec{b})^2-(\vec{a}\cdot\vec{p})^2+2\sqrt{(r_a^2(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2)(r_b^2(\vec{a}\times\vec{b})^2-(\vec{a}\cdot\vec{p})^2)} \tag{4.29}$$

moving the square root to one side we get:

$$(\vec{p}\cdot(\vec{a}\times\vec{b}))^2-(r_a^2+r_b^2)(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2-(\vec{a}\cdot\vec{p})^2 > 2\sqrt{(r_a^2(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2)(r_b^2(\vec{a}\times\vec{b})^2-(\vec{a}\cdot\vec{p})^2)} \tag{4.30}$$

which can only be true if the left side is positive, if it is then we can get rid of the last square root by squaring both sides again, this results in:

$$\big((\vec{p}\cdot(\vec{a}\times\vec{b}))^2-(r_a^2+r_b^2)(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2-(\vec{a}\cdot\vec{p})^2\big)^2 > 4\big(r_a^2(\vec{a}\times\vec{b})^2-(\vec{b}\cdot\vec{p})^2\big)(r_b^2(\vec{a}\times\vec{b})^2-(\vec{a}\cdot\vec{p})^2) \tag{4.31}$$

now as before $\vec{p}$ and r_a and r_b are linear functions of two parameters corresponding to either cone, but now we are dealing with a fourth order polynomial which makes finding local minima considerably more complicated.

4.4 Infinite Cones

Using a clever trick it is actually possible to provide a consistent model for cones with an infinite aspect ratio, even with a non-zero packing fraction. This method was first used to study the behaviour of spherocylinders [2, sec. VII-A] but can readily be adapted to other shapes.

The basic idea is to increase the length of the particles, while simultaneously changing their coordinates and orientation to (try to) prevent overlap. Using this it is possible to go back and forth between systems with very high and very low aspect ratios. However as the aspect ratio goes to infinity the distances go to infinity as well, this can be dealt with by using a different coordinate system. In fact, by picking the coordinate system in a clever way the position and orientation of each cone will remain constant in this (unphysical) coordinate system.

Definition

The coordinates we will use are given by $(x', y', z') = (\frac{x}{D}, \frac{y}{D}, \frac{z}{L})$ (where x, y, z are the Euclidean-space coordinates). These new coordinates will be referred to as the 'reduced' coordinates.

The position of the cone is represented by a point and its orientation can be represented by a line going through its axis. Transforming from one aspect ratio to another is completely determined by requiring that the positions and axes of these cones remain constant in these reduced coordinates.

The transformation can be divided into two parts, first the system is stretched (deforming some cones) and then all deformed cones are replaced with 'proper' cones with the same position and orientation. What this looks like in Cartesian coordinates can be seen in figure 8. In reduced coordinates the position and axis of each particle remains the same, but their shape changes (see figure 9), in this coordinate system it is fairly straightforward to continue the process to arbitrary aspect ratios.

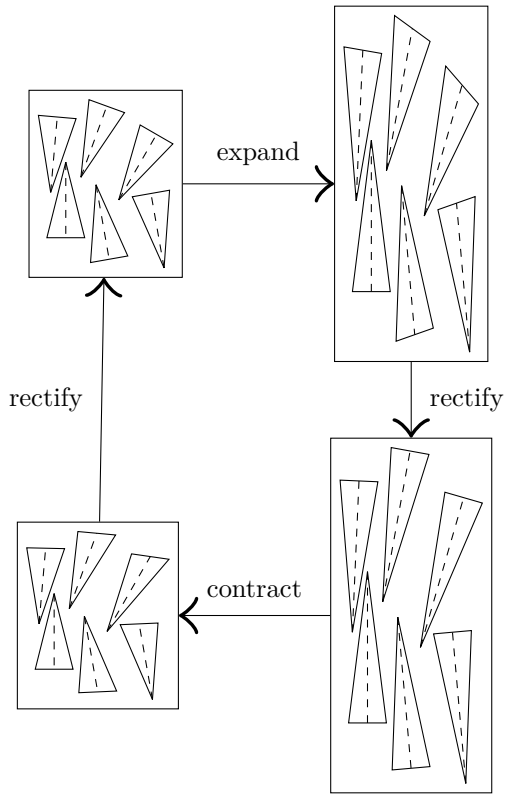


Figure 8: Changing the aspect ratio.

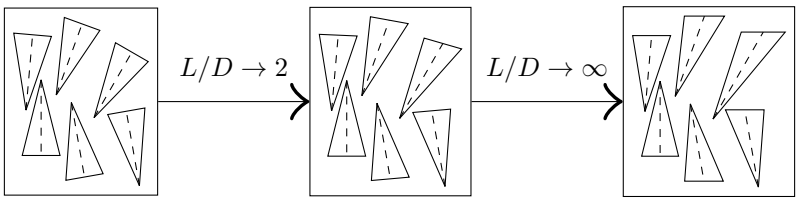


Figure 9: Changing the aspect ratio in reduced coordinates.

To make this more rigorous it makes sense to define a configuration of 'infinite' cones as an equivalence class C of configurations of (finite) cones, with the equivalence relation generated by our transformation. These configurations will be considered overlapping / non-overlapping if there is some aspect ratio AR such that *all* configurations $\gamma \in C$ with a higher aspect ratio overlap / do not overlap, respectively.

This is a somewhat abstract (though rigorous) description, but there are a few problems. There is no easy way to find out if a particular configuration of infinite cones is overlapping / non-overlapping, and there might be configurations that are neither. However as long as we restrict ourselves to finitely many cones this will not be a problem.

To show this, we shall try to find what a cone looks like in our reduced coordinate system as the aspect ratio goes to infinity. It should be obvious that two cones overlap in Cartesian coordinates if and only if they overlap in reduced coordinates, and since position and orientation remain fixed in reduced coordinates, there is a decent chance the shape of a cone might have some well defined 'limit' at $L/D = \infty$. This shape we will call the 'infinite' cone, and because of the way it is defined, the shapes of two 'infinite' cones will overlap if there is some aspect ratio L/D above which the corresponding system of two finite cones overlap. Conversely if the shapes of two 'infinite' cones do not overlap then there is some aspect ratio L/D above which the corresponding system of two finite cones does not overlap.

Hence, if this 'infinite' cone is well behaved, then it immediately suggests a way to detect if a configuration of (a finite number of) infinite cones is overlapping, and this will also prove that any configuration of infinite cones with a finite number of cones must either be overlapping or non-overlapping.

To describe the shape of the 'infinite' cone, let us start with a cone with some arbitrary aspect ratio, with its tip at the origin and orientation $\vec{n}$, which is normalized such that $n_{z'} = 1$ (the case where n_z is negative follows analogously). A point $\vec{u}$ is then inside this cone if it satisfies the following equations:

$$\begin{gathered} \left(\vec{n}\frac{\vec{n}\cdot\vec{u}}{\vec{n}^2} - \vec{u}\right)^2 \leq \left(\frac{\vec{n}\cdot\vec{u}}{2(L/D)|\vec{n}|}\right)^2 \\ 0 \leq \frac{\vec{n}\cdot\vec{u}}{L|\vec{n}|} \leq 1 \end{gathered} \tag{4.32}$$

where the inner product is the one induced by the original coordinate system i. e.

$$\vec{u}\cdot\vec{v} = D^2 u_{x'} v_{x'} + D^2 u_{y'} v_{y'} + L^2 u_{z'} v_{z'} \tag{4.33}$$

Now let us see what happens to the shape of the cone as the aspect ratio becomes arbitrarily high. For this we will first need to solve the following

$$0 \leq \lim_{L/D\to\infty} \frac{\vec{n}\cdot\vec{u}}{L|\vec{n}|} \leq 1. \tag{4.34}$$

where

$$\begin{aligned}
\lim_{L/D\to\infty} \frac{\vec{n}\cdot\vec{u}}{L|\vec{n}|} &= \lim_{L/D\to\infty} \frac{D^2 n_{x'}u_{x'} + D^2 n_{y'}u_{y'} + L^2 n_{z'}u_{z'}}{L\sqrt{D^2 n_{x'}^2 + D^2 n_{y'}^2 + L^2 n_{z'}^2}} \\
&= \lim_{L/D\to\infty} \frac{D^2 n_{x'}u_{x'} + D^2 n_{y'}u_{y'} + L^2 n_{z'}u_{z'}}{L^2\sqrt{\frac{D^2}{L^2} n_{x'}^2 + \frac{D^2}{L^2} n_{y'}^2 + n_{z'}^2}} \\
&= \lim_{L/D\to\infty} \frac{D^2 n_{x'}u_{x'} + D^2 n_{y'}u_{y'} + L^2 n_{z'}u_{z'}}{L^2|n_{z'}|} \\
&= \lim_{L/D\to\infty} \frac{D^2}{L^2} n_{x'}u_{x'} + \frac{D^2}{L^2} n_{y'}u_{y'} + n_{z'}u_{z'} \\
&= \lim_{L/D\to\infty} \frac{n_{z'}u_{z'}}{|n_{z'}|} \\
&= n_{z'}u_{z'}
\end{aligned} \tag{4.35}$$

since we are assuming $n_{z'}$ to be 1 this tells us $0 \le u_{z'} \le 1$. To simplify the other inequality we will need the following:

$$\lim_{L/D\to\infty} \frac{L\vec{n}}{|\vec{n}|} = \lim_{L/D\to\infty} \frac{L\vec{n}}{\sqrt{D^2 n_{x'}^2 + D^2 n_{y'}^2 + L^2 n_{z'}^2}} \tag{4.36}$$

$$= \lim_{L/D\to\infty} \frac{\vec{n}}{\sqrt{\frac{D^2}{L^2} n_{x'}^2 + \frac{D^2}{L^2} n_{y'}^2 + n_{z'}^2}} \tag{4.37}$$

$$= \frac{\vec{n}}{|n'_z|} = \vec{n}. \tag{4.38}$$

Combinging this with (4.32) we get:

$$\begin{aligned}
0 &\ge \lim_{L/D\to\infty} \left(\vec{n}\frac{\vec{n}\cdot\vec{u}}{\vec{n}^2} - \vec{u}\right)^2 - \left(\frac{\vec{n}\cdot\vec{u}}{2(L/D)|\vec{n}|}\right)^2 \\
&= \lim_{L/D\to\infty} \left(\frac{L\vec{n}}{|\vec{n}|}\frac{\vec{n}\cdot\vec{u}}{L|\vec{n}|} - \vec{u}\right)^2 - \left(\frac{D}{2}\frac{\vec{n}\cdot\vec{u}}{L|\vec{n}|}\right)^2 \\
&= (\vec{n}u_{z'} - \vec{u})^2 - \left(\frac{D}{2}\right)^2 u_{z'}^2 \\
&= D^2(n_{x'}u_{z'} - u_{x'})^2 + D^2(n_{y'}u_{z'} - u_{y'})^2 - (D/2)^2 u_{z'}^2
\end{aligned} \tag{4.39}$$

For a fixed $u_{z'}$ this describes a disk centred at $\vec{n}u_{z'}$ with radius $\frac{1}{2}u_{z'}$.

For an infinite cone with a tip at $\vec{t}$ and orientation $\vec{n}$ this is a disk centred at $\vec{t}+\vec{n}(u_{z'}-t_{z'})n_{z'}$ and a radius $\frac{1}{2}(u_{z'} - t_{z'})n_{z'}$. This simple shape makes it quite easy to check if two 'infinite cones' are overlapping.

This way of deriving the shape is somewhat tedious, yet rigorous. Later on we described an alternative way to derive the shape, provided we can find the shape for at least one orientation. This method will give the same result, confirming that the theory works as expected, and will make it easier to derive the 'infinite' shapes for other kinds of particles.

Volume

The volume of a regular cone is simply given by $\frac{\pi}{3}L\left(\frac{D}{2}\right)^2$ and it is easy to verify that this is also the volume of the shape of the infinite cone when considered in 'physical' coordinates. Forcing this value to stay finite does bring up some issues, which brings us to the next section.

Order Parameter

If we require the volume to stay finite then D becomes arbitrarily small, which in turn suggests that, since horizontal distances are measured units of D, and the angle with the nematic director is determined by these horizontal distances, every configuration of infinite cones is essentially perfectly aligned with the nematic director.

This can be made stronger by considering the value of $\cos(\theta)$ where θ is the angle with the nematic director. For a cone with orientation $\vec{n}$ and with the nematic director $\vec{d}$ pointing along the z-axis this is given by:

$$\cos(\theta) = \frac{\vec{n}\cdot\vec{d}}{|\vec{n}|} = \frac{Ln_{z'}}{\sqrt{D^2n_{x'} + D^2n_{y'} + L^2n_{z'}}} \tag{4.40}$$

which in the limit $L/D \to \infty$ just becomes 1, indicating that no matter how the infinite cone is oriented, with a high enough aspect ratio it can be aligned arbitrarily well with the nematic director.

It is however more interesting to consider the value of the 'infinitesimal angle' Θ, which for a nematic director pointing along the z-axis is defined as follows:

$$\Theta = \frac{L}{D}\sin(\theta) = \frac{L}{D}\sqrt{1 - \left(\frac{Ln_{z'}}{\sqrt{D^2n_{x'}^2 + D^2n_{y'}^2 + L^2n_{z'}^2}}\right)^2} \tag{4.41}$$

$$= \frac{L}{D}\sqrt{\frac{D^2n_{x'}^2 + D^2n_{y'}^2 + L^2n_{z'}^2 - L^2n_{z'}}{D^2n_{x'}^2 + D^2n_{y'}^2 + L^2n_{z'}^2}} \tag{4.42}$$

$$= \frac{L}{D}\sqrt{\frac{D^2n_{x'}^2 + D^2n_{y'}^2}{D^2n_{x'}^2 + D^2n_{y'}^2 + L^2n_{z'}^2}} \tag{4.43}$$

$$= \sqrt{\frac{n_{x'}^2 + n_{y'}^2}{\frac{D^2}{L^2}n_{x'}^2 + \frac{D^2}{L^2}n_{y'}^2 + n_{z'}^2}}. \tag{4.44}$$

Note that this can be related to the nematic order parameter as follows:

$$\left\langle\Theta^2\right\rangle = \left\langle\left(\frac{L}{D}\right)^2\sin(\theta)^2\right\rangle = \left(\frac{L}{D}\right)^2\left\langle 1-\cos(\theta)^2\right\rangle = \frac{2}{3}\left(\frac{L}{D}\right)^2(1-S). \tag{4.45}$$

Hence, $\left\langle\Theta^2\right\rangle$ seems a good candidate for an order parameter. In the limit $L/D \to \infty$ the value Θ^2 simplifies as follows:

$$\lim_{L/D\to\infty}\frac{n_{x'}^2 + n_{y'}^2}{\frac{D^2}{L^2}n_{x'}^2 + \frac{D^2}{L^2}n_{y'}^2 + n_{z'}^2} = \frac{n_{x'}^2 + n_{y'}^2}{n_{z'}^2} = n_{x'}^2 + n_{y'}^2 \tag{4.46}$$

where the last step assumes we have normalized such that $n_{z'} = \pm 1$.

Now, there are cases where the nematic director $\vec{d}$ is not pointing exactly along the z-axis (in reduced coordinates). The most natural way of solving this is by first rotating the system such that $\vec{d}$ points along the z-axis (more detailed information on how to rotate in reduced coordinates will follow shortly), and then use the above formula. This results in the value:

$$\left\langle (n_{x'} - d_{x'})^2 + (n_{y'} - d_{y'})^2 \right\rangle. \tag{4.47}$$

Using relation (4.45) and the definition of the nematic director as the vector maximizing the nematic order parameter, we see that $\vec{d}$ should be chosen to minimize the above expression. Which results in $d_{x'} = \langle n_{x'} \rangle$ and $d_{y'} = \langle n_{x'} \rangle$. With a slight abuse of notation it therefore makes sense to define the new order parameter as follows:

$$\mathrm{Var}(\Theta) = \left\langle (n_{x'} - d_{x'})^2 + (n_{y'} - d_{y'})^2 \right\rangle \tag{4.48}$$

with the following relation to the nematic order parameter:

$$\mathrm{Var}(\Theta) = \frac{2}{3}\left(\frac{L}{D}\right)^2 (1 - S). \tag{4.49}$$

A similar order parameter has also been considered for the infinite spherocylinders, which this method is derived from [2]. The definition of $\mathrm{Var}(\Theta)$ is slightly different in that its definition makes sense for both the finite cones and the infinite cones, but this is more of a technicality.

The advantage of using $\mathrm{Var}(\Theta)$ over the standard nematic order parameter is that, if cones indeed behave like infinite cones as at high aspect ratios, then $\mathrm{Var}(\Theta)$ will converge onto the same value (since for long enough aspect ratios the value of $\mathrm{Var}(\Theta)$ does not change much when changing the aspect ratio) and can be compared directly between configurations of different aspect ratios, and the infinite cones.

Rotations in Reduced Coordinates

By reusing some of the above calculations it is also possible to describe how to rotate in reduced coordinates and derive the shape of the infinite cone in an easier way. Consider for instance a simple rotation in the y-z plane:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) \\ 0 & -\sin(\theta) & \cos(\theta) \end{pmatrix} \tag{4.50}$$

the corresponding transformation in reduced coordinates looks as follows:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & \frac{L}{D}\sin(\theta) \\ 0 & -\frac{D}{L}\sin(\theta) & \cos(\theta) \end{pmatrix} \tag{4.51}$$

now if we start with an orientation $\vec{n} = (0, 0, 1)$, rotate it with the above matrix (in either coordinate system) and increase the aspect ratio while keeping the orientation fixed, then as we just saw, the value $\frac{L}{D}\sin(\theta)$ becomes equal to $\sqrt{n_{x'}^2 + n_{y'}^2}$, the value $\cos(\theta)$ becomes

1 and $-\frac{D}{L}\sin(\theta)$ goes to 0. The matrix we end up in the infinite aspect ratio limit is then given by:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & n_{y'} \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.52}$$

Doing the same to a rotation in the x-z plane and multiplying the two yields the following matrix for rotating something from $\vec{n} = (0, 0, 1)$ to $\vec{n} = (n_{x'}, n_{y'}, 1)$:

$$\begin{pmatrix} 1 & 0 & n_{x'} \\ 0 & 1 & n_{y'} \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.53}$$

Starting with a regular cone with orientation $\vec{n} = (0, 0, 1)$ and 'rotating' it with the above matrix indeed yields a cone with the exact same shape as the 'infinite' cone. To get all different orientations, even for shapes without cylindrical symmetry, the above can be combined with a rotation in the x-y plane, yielding the following matrix:

$$\begin{pmatrix} \cos(\phi) & \sin(\phi) & n_{x'} \\ -\sin(\phi) & \cos(\phi) & n_{y'} \\ 0 & 0 & 1 \end{pmatrix}. \tag{4.54}$$

Using these transformations it is now easy to derive *all* different orientations of any 'infinite' shape, as long as we know the shape for one particular orientation. In some cases changing the aspect ratio does not affect the shape (in reduced coordinates) of a particle pointing along the z-axis, this is the case for cones, ellipsoidal particles, and cylinders. Hence it is trivial to derive the shape of the 'infinite' version of those shapes for that particular orientation. For spherocylinders the spherical cap on top of the cylinders presents a small problem, but since this cap becomes negligibly small for high aspect ratios they can be considered cylinders for the purposes of this method. Similarly, the 'infinite' ice cream cone is the same as the 'infinite' cone.

Overlap Detection

Even though the method could in principle work for arbitrary shapes, infinite cones do have the distinct advantage that there exists a relatively simple way to check if two of them overlap.

The cross-section of two infinite cones at some constant z' reduces to two disks, and the position and radius of these disks both change linearly with z'. So to find if they intersect it is enough to check if the distance between the centres of the disks is less than the sum of their radii. Since the positions change linearly, the square of the distance changes quadratically with respect to z'. Obviously the squared sum of the radii also changes quadratically so finding an intersection simply reduces to finding the root of a quadratic equation in some interval.

In fact, consider an infinite cone with its tip at the origin, pointing directly upwards and an infinite cone with its tip at $\vec{p}$ and an orientation $\vec{n}$. At a fixed z' the distance between the centres of the disks is

$$\left| p_{x'y'} + (z' - p_{z'}) \frac{n_{x'y'}}{n_{z'}} \right|. \tag{4.55}$$

Here the notation $v_{x'y'}$ is meant to indicate the projection of v onto the x'-y' plane. This expression can also be written as $\left|\vec{d} + z' \frac{\vec{n}}{n_{z'}}\right|_{x'y'}$ where $\vec{d} = \vec{p} - p_{z'} \frac{\vec{n}}{n_{z'}}$. The two radii are simply $\frac{1}{2} z'$ and $\frac{1}{2}(z' - p_{z'}) n_{z'}$.

The two cones then intersect if there is a z' such that:

$$\left(\vec{d} + z' \frac{\vec{n}}{n_{z'}}\right)^2_{x'y'} - \left(\tfrac{1}{2} z' + \tfrac{1}{2}(z' - p_{z'}) n_{z'}\right)^2 < 0 \tag{4.56}$$

with $z' \in [0, 1] \cap [p_{z'}, p_{z'} + 1]$. To find if this is the case it is enough to check the boundaries of this interval, and the minimum (if it is inside the interval). If $n_{z'} = -1$ then the sum of the two radii is $\frac{1}{2} p_{z'}$ which is constant, hence the minimum is at:

$$z' = \frac{d_{x'y'} \cdot n_{x'y'}}{n^2_{x'y'}} \tag{4.57}$$

Otherwise the sum of the two radii is $z' - \frac{1}{2} p_{z'}$ and the minimum is at:

$$z' = \frac{\frac{1}{2} p_{z'} - d_{x'y'} \cdot n_{x'y'}}{n^2_{x'y'} - 1} \tag{4.58}$$

note that this is only a minimum if the denominator is positive.

This is enough information to detect overlap between arbitrary cones, since translations and 'rotations' do not affect equation (4.55) as long as we replace $\vec{p}$ and $\vec{n}$ with the relative position and orientation. Explicitly, for two arbitrary cones with positions $\vec{q_a}$ and $\vec{q_b}$ and orientations a and b overlap can be detected with the above formulas with $n_{x'y'} = (\vec{a} - \vec{b})_{x'y'}$, $n_{z'} = a_{z'} b_{z'}$, and $\vec{p} = \vec{q_a} - \vec{q_b}$. This results in algorithm 8.

Algorithm 8 Infinite Cone Overlap Detection.

function InfiniteConeOverlap($\vec{p}, a, b$)
  $l \leftarrow \max(0, p_{z'})$
  $h \leftarrow \min(1, p_{z'} + 1)$
  $n_{x'y'} \leftarrow (\vec{a} - \vec{n})_{x'y'}$
  $n_{z'} \leftarrow a_{z'} b_{z'}$
  $\vec{d} \leftarrow \vec{p} - p_{z'} \frac{\vec{n}}{n_{z'}}$
  if $l > h$ **then**
    return false
  if $n_{z'} = -1$ **then**
    $z' \leftarrow \frac{d_{x'y'} \cdot n_{x'y'}}{n^2_{x'y'}}$
    $z' \leftarrow \max(l, \min(h, z'))$
    return $\left(\vec{d} - z'\vec{n}\right)^2_{x'y'} < \left(\frac{1}{2} p_{z'}\right)^2$
  else
    $D \leftarrow n^2_{x'y'} - 1$
    if $D > 0$ **then**
      $z' \leftarrow \frac{\frac{1}{2} p_{z'} - d_{x'y'} \cdot n_{x'y'}}{D}$
      $z' \leftarrow \max(l, \min(h, z'))$
      return $\left(\vec{d} + z'\vec{n}\right)^2_{x'y'} < \left(z' - \frac{1}{2} p_{z'}\right)^2$
    else
      for all $z' \in \{l, h\}$ **do**
        if $\left(\vec{d} + z'\vec{n}\right)^2_{x'y'} < \left(z' - \frac{1}{2} p_{z'}\right)^2$ **then**
          return true
      return false

4.5 Double Cones

An easy generalisation of the previous shapes is the so-called 'double cone', which is the shape created by two cones glued together at their tips as in figure 10. This can also be done to the ice cream cones. Even though these shapes are no longer convex they can be treated with virtually the same techniques. One reason these kind of shapes are interesting is the strong dependence of the mutually excluded volume on the angle between the two cones.

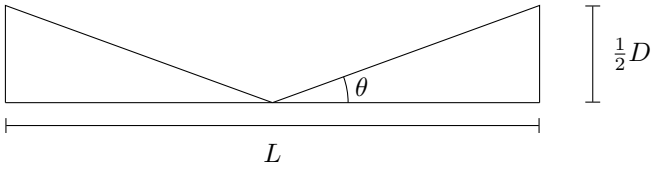


Figure 10: Construction of a double cone.

A double cone of a certain aspect ratio L/D is the result of gluing two single cones with an aspect ratio $L/(2D)$. Volume is simply twice the volume of the single cones and overlap detection between two double cones can be done by checking all pairs of single cones (except the ones belonging to the same double cone). For the double ice cream cones and double infinite cones it is also possible to generalize the algebraic methods used to detect overlap.

Double Ice Cream Cones

The double ice cream cone is constructed in a way similar to how the double cone is constructed from the single cone. Again most properties can simply be derived from the single ice cream cones. Except for the overlap detection. However it is not particularly difficult to modify the method to work for double ice cream cones as well.

For finding overlap between two ice cream cones we needed to find a minimum of the expression:

$$((\vec{x}-\vec{y})+t\vec{a}-s\vec{b})^2-\gamma^2(t+s)^2 \tag{4.59}$$

if we allow t and s to become negative (to allow for double cones) then the radii of the spheres we are dealing with should be $\gamma|t|$ instead of γt, which means we should really be looking at the expression:

$$((\vec{x}-\vec{y})+t\vec{a}-s\vec{b})^2-\gamma^2(|t|+|s|)^2. \tag{4.60}$$

now the expression $|t|+|s|$ is either equal to $|t+s|$ or $|t-s|$ depending on whether t and s have the same sign. It is also larger than either of them. This means that it is enough to check if either of the following expressions drops below 0

$$((\vec{x}-\vec{y})+t\vec{a}-s\vec{b})^2-\gamma^2(t+s)^2 \tag{4.61}$$

$$((\vec{x}-\vec{y})+t\vec{a}-s\vec{b})^2-\gamma^2(t-s)^2. \tag{4.62}$$

Writing $\bar{s}=-s$ the latter can be rewritten as

$$((\vec{x}-\vec{y})+t\vec{a}-\bar{s}(-\vec{b}))^2-\gamma^2(t+\bar{s})^2 \tag{4.63}$$

which has the same form as (4.59) except $\vec{b}$ is replaced with $-\vec{b}$. This means that checking for collisions can be done using almost the same method as before, except we need to change the

range of the parameters from $[0, 2]$ to $[-2, 2]$ and repeat the check with one of the orientations reversed. A way to accomplish this using a generalized IccOverlap method is described in algorithm 9. For maximum performance it might be better to write out all expressions explicitly in terms of $\vec{p}^2, \vec{b} \cdot \vec{p}, \vec{a} \cdot \vec{b}$ and $\vec{a} \cdot \vec{p}$ since a lot of calculations could probably be reused.

Algorithm 9 Double Ice Cream Cone Overlap Detection.

function ParametricIccOverlap($\vec{p}, \vec{a}, \vec{b}, l, h$)
  $M \leftarrow \begin{pmatrix} 1-\gamma^2 & -\vec{a}\cdot\vec{b}-\gamma^2 & \vec{a}\cdot\vec{p} \\ -\vec{a}\cdot\vec{b}-\gamma^2 & 1-\gamma^2 & -\vec{b}\cdot\vec{p} \\ \vec{a}\cdot\vec{p} & -\vec{b}\cdot\vec{p} & \vec{p}^2 \end{pmatrix}$
  $\Delta \leftarrow (1-\gamma^2)^2 - (\vec{a}\cdot\vec{b}+\gamma^2)^2$
  $(t, s) \leftarrow \Delta^{-1} \begin{pmatrix} 1-\gamma^2 & \vec{a}\cdot\vec{b}+\gamma^2 \\ \vec{a}\cdot\vec{b}+\gamma^2 & 1-\gamma^2 \end{pmatrix} \begin{pmatrix} -\vec{a}\cdot\vec{p} \\ \vec{b}\cdot\vec{p} \end{pmatrix}$
  if $\Delta > 0$ **and** $(t, s) \in [l, h] \times [l, h]$ **and** $(t, s, 1)M(t, s, 1)^T < 0$ **then**
    return true
  for all $s \in \{-l, h\}$ **do**
    $t \leftarrow \frac{s(\vec{a}\cdot\vec{b}+\gamma^2)-\vec{a}\cdot\vec{p}}{1-\gamma^2}$
    $t \leftarrow \max(l, \min(h, t))$
    if $(t, s, 1)M(t, s, 1)^T < 0$ **then**
      return true
  for all $t \in \{-l, h\}$ **do**
    $s \leftarrow \frac{t(\vec{a}\cdot\vec{b}+\gamma^2)+\vec{b}\cdot\vec{p}}{1-\gamma^2}$
    $s \leftarrow \max(l, \min(h, s))$
    if $(t, s, 1)M(t, s, 1)^T < 0$ **then**
      return true
  return false
function DoubleIccOverlap($\vec{p}, \vec{a}, \vec{b}$)
  if ParametricIccOverlap($\vec{p}, \vec{a}, \vec{b}, -2, 2$) **then**
    return true
  else if ParametricIccOverlap($\vec{p}, \vec{a}, -\vec{b}, -2, 2$) **then**
    return true
  return false

Double Infinite Cones

For the infinite cones the quantity that determines whether they intersect is:

$$\left(\vec{d} + z' \frac{n}{n_{z'}}\right)^2_{x'y'} - \left(\tfrac{1}{2}z' + \tfrac{1}{2}(z' - p_{z'})n_{z'}\right)^2 \tag{4.64}$$

which, again needs to be modified slightly to allow for negative values of z' and $(z' - p_{z'})$, resulting in:

$$\left(\vec{d} + z' \frac{n}{n_{z'}}\right)^2_{x'y'} - \left(|\tfrac{1}{2}z'| + |\tfrac{1}{2}(z' - p_{z'})|\right)^2. \tag{4.65}$$

This means that to check for overlap we should check whether either of the following equations drops below 0

$$\begin{aligned} &\left(\vec{d} + z'n\right)^2_{x'y'} - \left(z' - \tfrac{1}{2}p_{z'}\right)^2 \\ &\left(\vec{d} - z'n\right)^2_{x'y'} - \left(\tfrac{1}{2}p_{z'}\right)^2. \end{aligned} \tag{4.66}$$

Note that these are precisely the two equations corresponding to the two different orientations of infinite cones, which means we can detect overlap with only some minor changes to algorithm 8 on page 33, resulting in algorithm 10.

Algorithm 10 Double Infinite Cone Overlap Detection.

function DoubleInfiniteConeOverlap($\vec{p}, \vec{a}, \vec{b}$)
  $l \leftarrow \max(-1, p_{z'} - 1)$
  $h \leftarrow \min(1, p_{z'} + 1)$
  $n_{x'y'} \leftarrow (\vec{a} - \vec{n})_{x'y'}$
  $n_{z'} \leftarrow a_{z'} b_{z'}$
  $\vec{d} \leftarrow \vec{p} - p_{z'} \frac{\vec{n}}{n_{z'}}$
  if $l > h$ **then**
    return false
  $z' \leftarrow \frac{\vec{d} \cdot n_{x'y'}}{n^2_{x'y'}}$
  $z' \leftarrow \max(l, \min(h, z'))$
  if $\left(\vec{d} - z'n\right)^2_{x'y'} < \left(\frac{1}{2}p_{z'}\right)^2$ **then**
    return true
  $D \leftarrow n^2_{x'y'} - 1$
  if $D > 0$ **then**
    $z' \leftarrow \frac{\frac{1}{2}p_{z'} - \vec{d} \cdot n_{x'y'}}{D}$
    $z' \leftarrow \max(l, \min(h, z'))$
    if $\left(\vec{d} + z'n\right)^2_{x'y'} < \left(z' - \frac{1}{2}p_{z'}\right)^2$ **then**
      return true
  else
    for all $z' \in \{l, h\}$ **do**
      if $\left(\vec{d} + z'n\right)^2_{x'y'} < \left(z' - \frac{1}{2}p_{z'}\right)^2$ **then**
        return true
  return false

5 Results

5.1 Single Cones

Determining the phase behaviour of the single cones was done using simulations of roughly 2000 particles, for 5 million Monte Carlo steps, using periodic boundary conditions and starting with the close packing of cones described in 4.2 on page 18. For comparison the same settings were used to run simulations of the infinite cones. These simulations were done for several reduced pressures starting at $\beta P v_p = 1$ and ending at $\beta P v_p = 15$ with steps of 1, for aspect ratios running from $L/D = 0.5$ to $L/D = 16$.

The equations of state resulting from these simulations are shown in figure 11 (refer to figure 20 on page 45 for individual plots). For lower aspect ratios $L/D < 4$ these equations of state show a clear jump at $\beta P v_p \approx 9$. For higher aspect ratios $L/D \geq 4$ there are two jumps one at $\beta P v_p \approx 12$ and another which seems to occur earlier as the aspect ratio gets bigger.

The equation of state of the infinite cones also agree well with the ones for the longer cones, especially at high densities. This is the first indication that the theory behind the infinite cones is successful in approximating the behaviour of long cones.

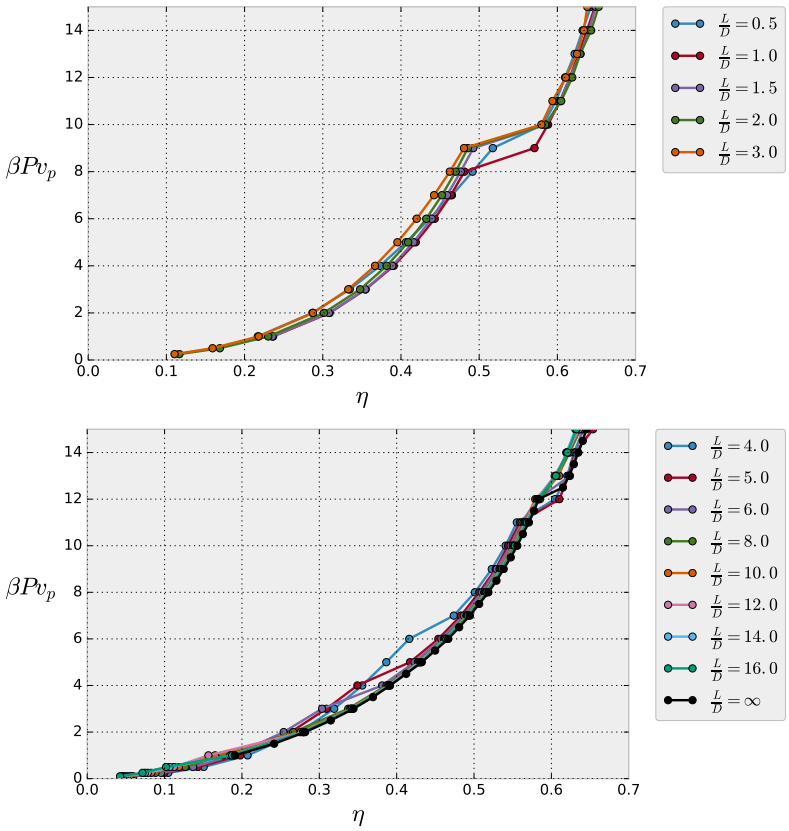
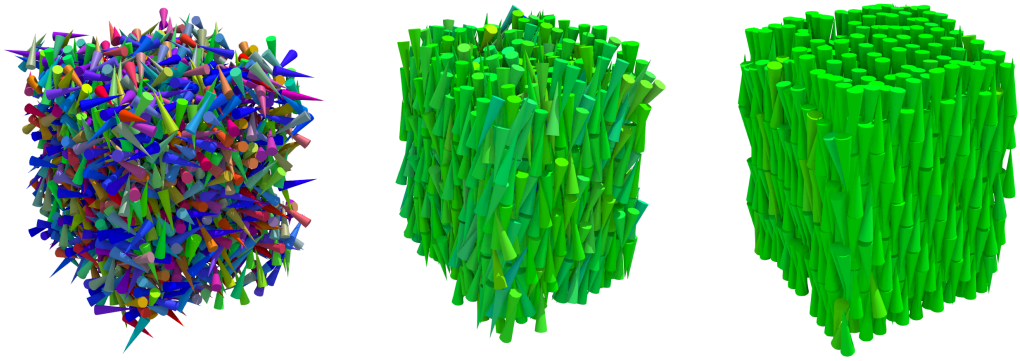


Figure 11: Equations of state, reduced pressure $\beta P v_0$ vs. volume fraction η, for single cones.

To get some idea of which phases there are, figure 12 shows some configurations for $L/D = 4$ after $5 \cdot 10^6$ Monte Carlo steps. In these three snapshots it is possible to recognise the isotropic and nematic phases, and a crystal phase. Curiously the smectic phase seems to be absent. This suggests that the two phase transitions that can be seen in the equation of state for longer cones correspond to the isotropic-nematic and the nematic-crystal phase transition. For shorter cones there is only one phase transition, a isotropic-crystal phase transition.



(a) Snapshot of cones at $\beta P v_p = 1$

(b) Snapshot of cones at $\beta P v_p = 8$

(c) Snapshot of cones at $\beta P v_p = 14$

Figure 12: Snapshots after $5 \cdot 10^6$ Monte Carlo steps of cones with $L/D = 4$ at different pressures.

If a smectic phase exists it must occur at a density lower than $\frac{2}{3}\frac{\pi}{2\sqrt{3}} \approx 60\%$. In a smectic phase the cones are aligned and confined to layers, which means that the bases of cones are forced to lie in the same planes. Since their bases are circular they cannot occupy more than $\frac{\pi}{2\sqrt{3}}$ of the area of each of these planes, and since each layer has two of these planes, and the volume of a cone is $1/3$ times its length times the area of its base, cones in a layered configuration cannot occupy significantly more than $\frac{2}{3}\frac{\pi}{2\sqrt{3}} \approx 60\%$ of the volume, without a significant change in structure.

The order parameters should give us some more detailed information on the exact behaviour of the cones, including the existence or non-existence of a smectic phase.

The nematic order parameter (figure 13) shows a clear jump at the first phase transition, indicating that the configuration is indeed isotropic at lower densities and becomes aligned when the density is high enough. Using equation (4.49) it is possible to predict the value of the nematic order parameter from the infinite cone simulation. For high densities this agrees well, further indication that cones with high aspect ratio start to behave like infinite cones.

To confirm our claim that the values of $\mathrm{Var}(\Theta)$ should converge onto the same if long cones start to behave like infinite cones, we can again use formula (4.49) to calculate $\mathrm{Var}(\Theta)$ from the nematic order parameters, the results of these calculations can be seen in figure 14. As expected the values all converge onto the same curve (which can be approximated remarkably well by a simple power law), especially in the nematic phase.

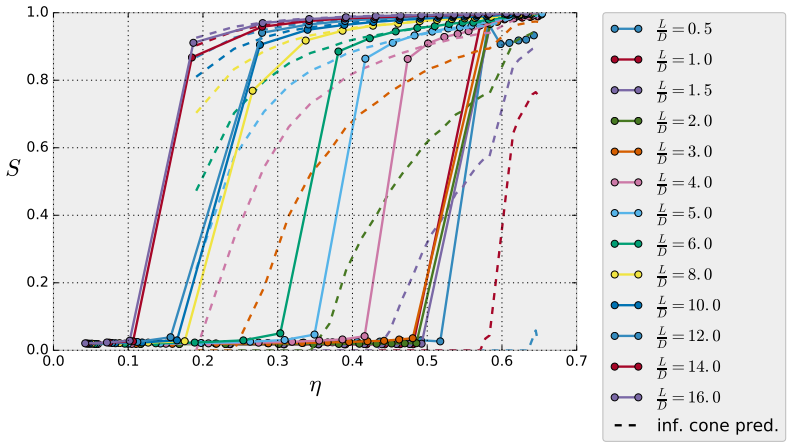


Figure 13: Nematic order parameter vs density of single cones. The dashed lines are predictions based on the value of $\mathrm{Var}(\theta)$ of the infinite cones.

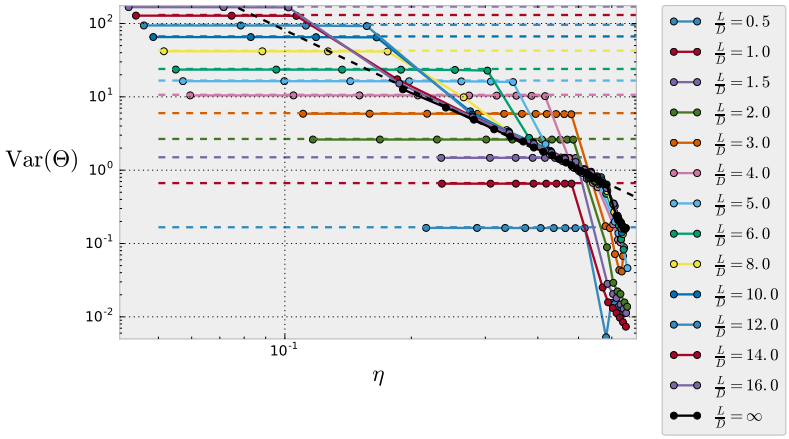
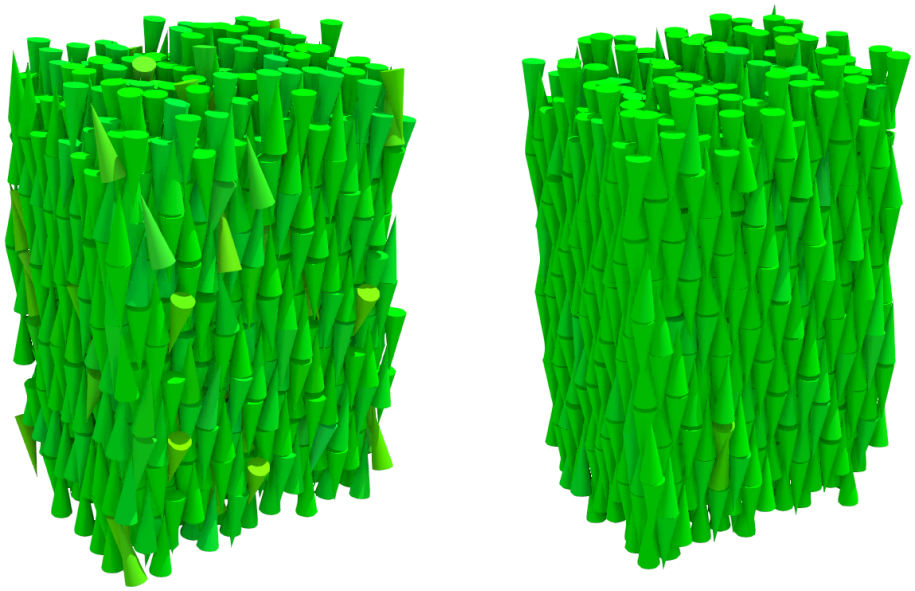


Figure 14: $\mathrm{Var}(\Theta)$ vs density of single cones. The dashed lines correspond to $S = 0$ for the finite cones, and $\mathrm{Var}(\Theta) = 0.153\eta^{-2.72}$ for the infinite cones.

The crystal order parameters (see figure 16) also show a jump, in at least two directions, indicating that the high density phase is indeed a crystal, and not a smectic phase. For the shorter cones this jump coincides with the jump in the nematic order parameter, but for the longer cones this is not the case. Furthermore these order parameters are not particularly stable for higher aspect ratios and drop down to 0 for high densities. This suggests that the crystal is distorted and that the initial configuration might not be the most stable crystal (at least not for all pressures). This is also confirmed by the order parameter ζ_Z (figure 16c), which shows almost no significant periodicity along the z-axis for higher aspect ratios, even though the initial condition *is* periodic along the z-axis.

Interestingly for $L/D = 3$ the order parameter ζ_Z first shows a significant jump, but then drops down again. Examining the corresponding snapshots (see figure 15) shows two clearly different crystal structures. The crystal structure with the lower density is most similar to the initial configuration, and corresponds to the peak in ζ_Z. Since it is unlikely the system would get stuck at low pressures but not at higher pressures, this suggests that the initial configuration *is* the most stable for lower densities at $L/D = 3$.

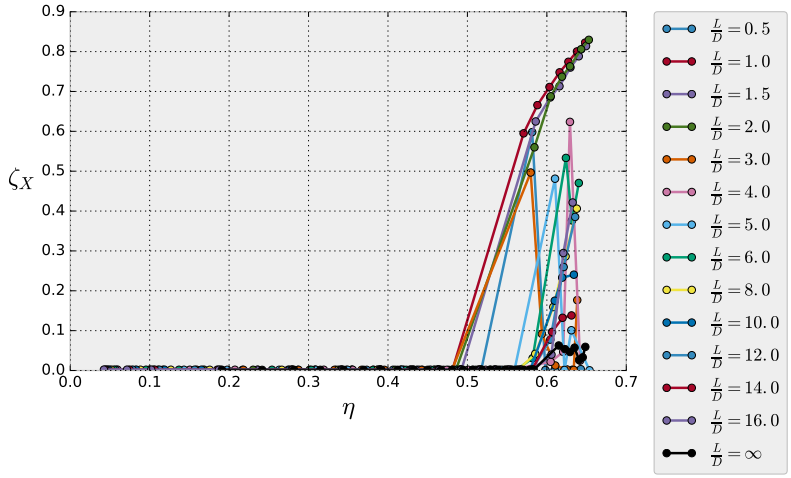


(a) Snapshot of cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 8$

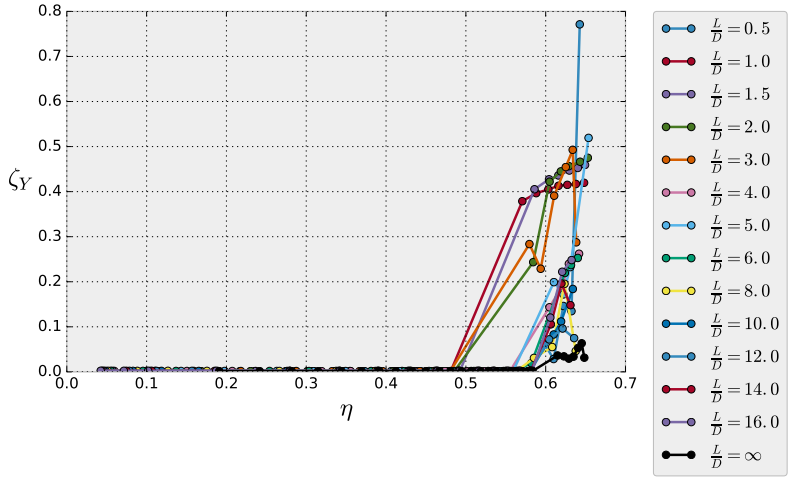
(b) Snapshot of cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 14$

Figure 15: Snapshots of crystal configurations of cones with $L/D = 3$.

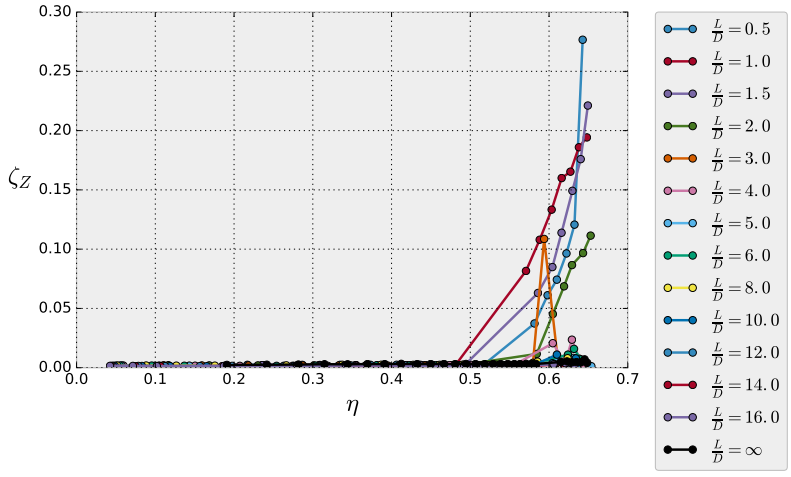
This hints at the following explanation for the instability of the crystal order parameters. Namely that the initial configuration is only stable for lower aspect ratios or low densities. At higher aspect ratios the initial configuration is only the preferred crystal for lower densities, hence why the crystal order parameters do jump at the phase transition, but then immediately drop down again as the system moves beyond the narrow range of densities where the initial condition is favourable as it either tries to form the other crystal phase. The imperfections caused by restructuring the crystal, or the stresses caused by being stuck in an unstable state deform the crystal enough to break the periodicity.



(a) Crystal order parameter along the x-axis ζ_X vs density for the single cones.



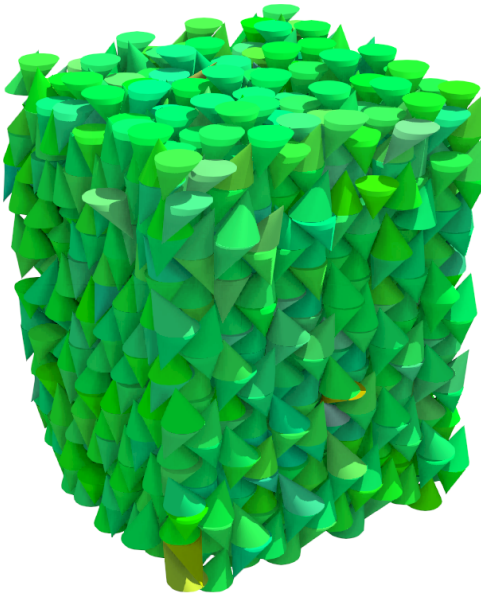
(b) Crystal order parameter along the y-axis ζ_Y vs density for the single cones.



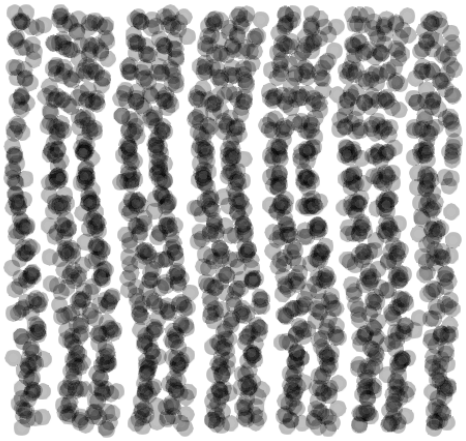
(c) Crystal order parameter along the z-axis ζ_Z vs density for the single cones.

Figure 16: Crystal order parameters

There is also some other interesting behaviour that can be seen for the crystal phase at high aspect ratios. This behaviour is most clearly visible for the infinite cones (figure 17). It seems the layers in the crystal like to pair up. Inside these layer-pairs the cones rotate slightly towards the outer edge of the layer-pairs. This way the edges of the cones on the outside of these layer pairs are aligned with the z-axis. Presumably this makes it easier for these layer-pairs to slide alongside one another (in the z-direction), which provides another explanation for the unexpectedly low values of ζ_Z.



(a) Snapshot of infinite cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 15.5$



(b) Particle positions of infinite cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 15.5$ viewed from above.

Figure 17: Infinite cone configuration after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 15.5$.

Even though the crystal order parameters do not always clearly detect the different kinds of crystal, they can still be used to find the location of the nematic-crystal phase transition, and combined with the nematic order parameter it is possible to construct a phase diagram (figure 18). To include the infinite cones it is more convenient to use D/L instead of L/D, which results in figure 19.

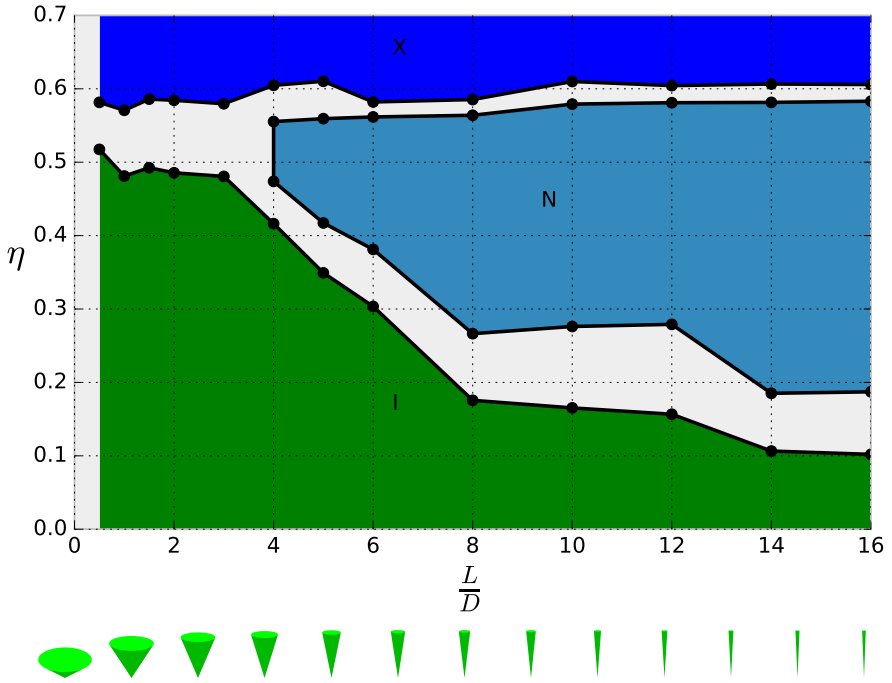


Figure 18: Phase diagram of the single cones.

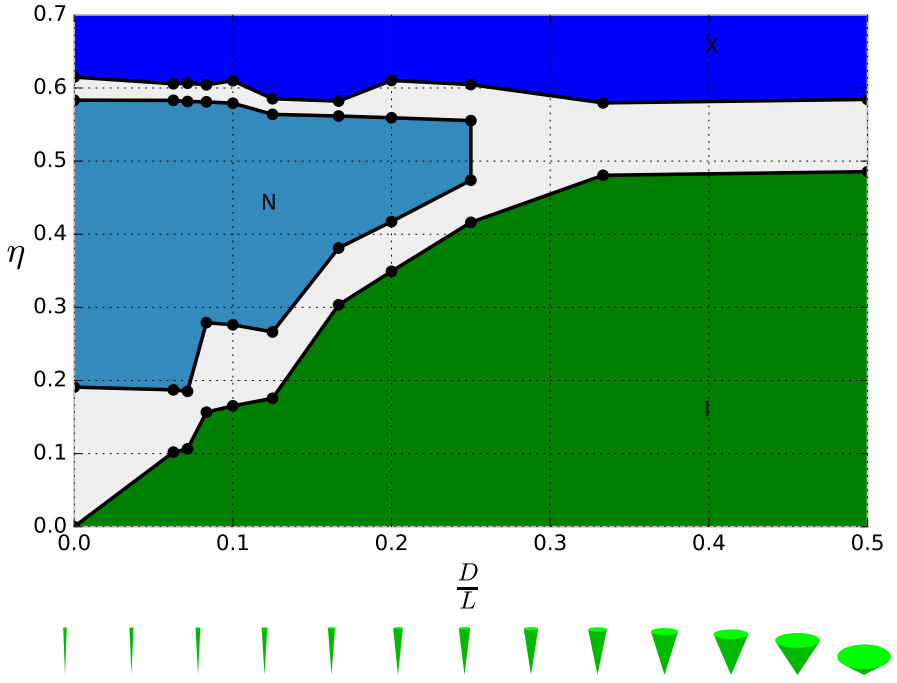
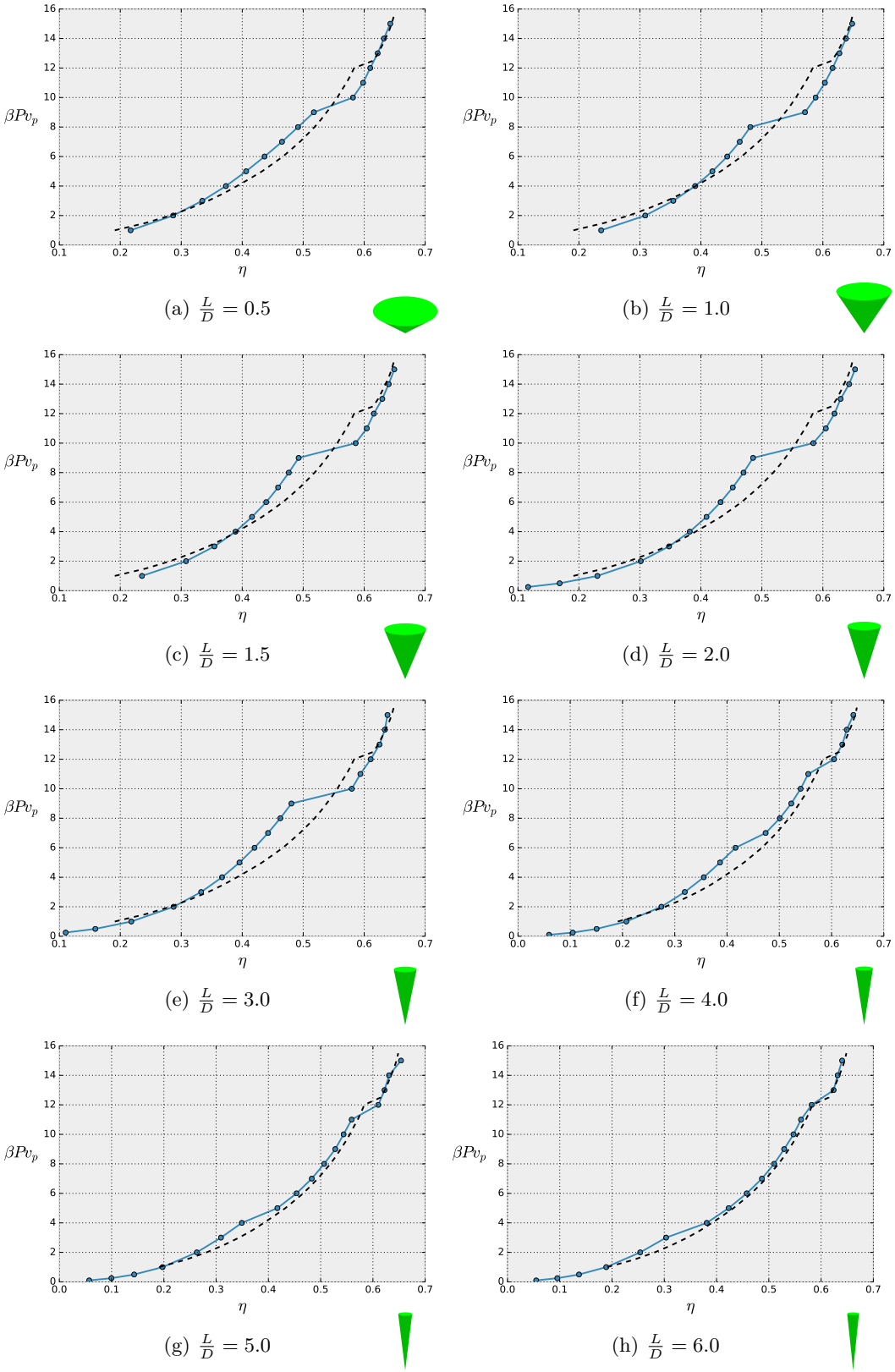


Figure 19: Flipped phase diagram of the single cones.



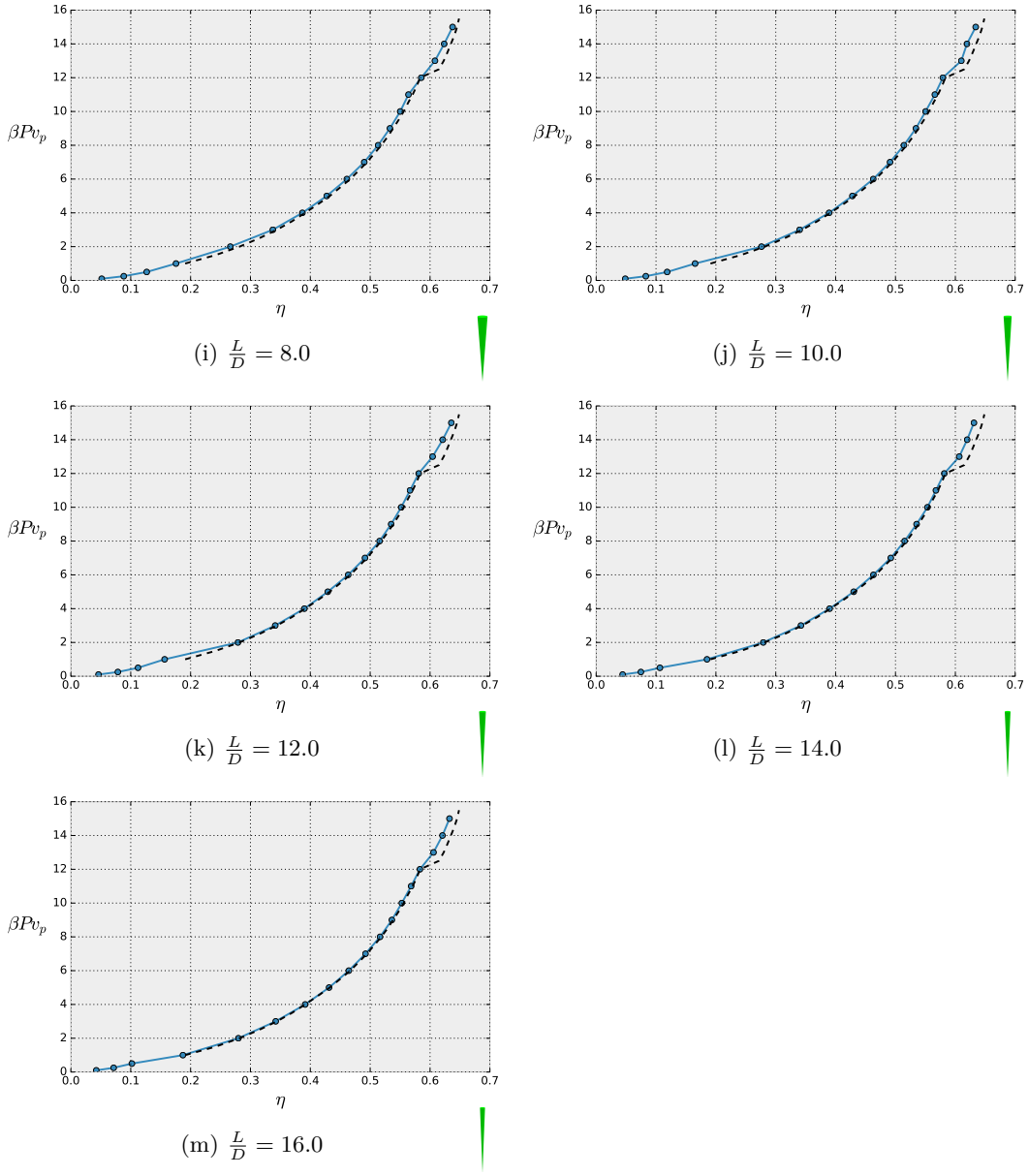
(a) $\frac{L}{D} = 0.5$ (b) $\frac{L}{D} = 1.0$

(c) $\frac{L}{D} = 1.5$ (d) $\frac{L}{D} = 2.0$

(e) $\frac{L}{D} = 3.0$ (f) $\frac{L}{D} = 4.0$

(g) $\frac{L}{D} = 5.0$ (h) $\frac{L}{D} = 6.0$

Figure 20: Individual equations of state for single cones, with the infinite cones for comparison



(i) $\frac{L}{D} = 8.0$ (j) $\frac{L}{D} = 10.0$

(k) $\frac{L}{D} = 12.0$ (l) $\frac{L}{D} = 14.0$

(m) $\frac{L}{D} = 16.0$

Figure 20: Individual equations of state for single cones, with the infinite cones for comparison

5.2 Ice Cream Cones

The ice cream cones were simulated with similar settings as the single cones, i. e. roughly 2000 particles, for 5 million Monte Carlo steps, using periodic boundary conditions, for the same pressures, and with aspect ratios from $L/D = 2$ to $L/D = 16$. The initial condition was generated by starting with the initial configuration of single cones and replacing those with the biggest inscribing ice cream cone. For the infinite aspect ratio limit we can reuse the simulations for the infinite cones, since the volume of the spherical cap goes to 0 in the infinite aspect ratio limit.

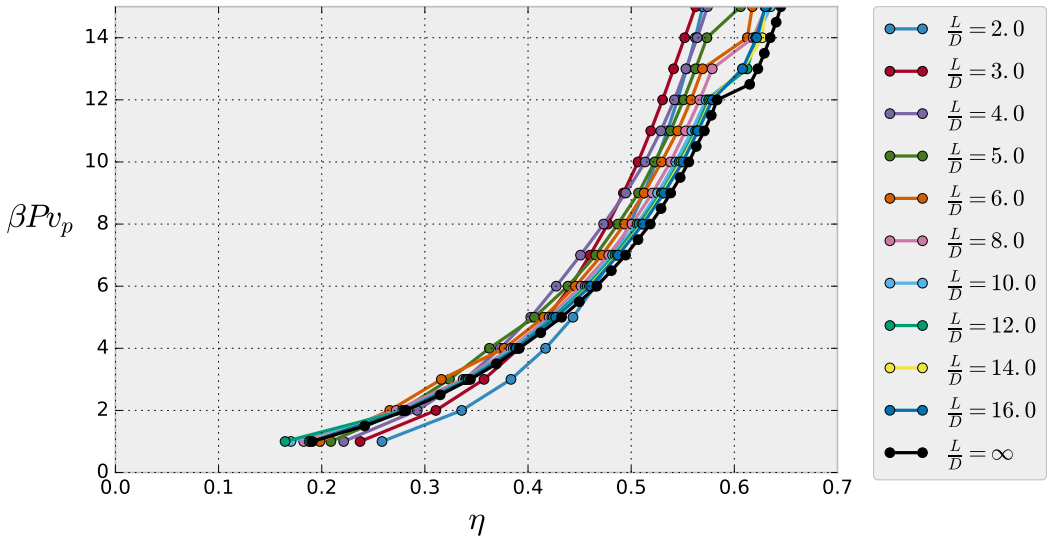


Figure 21: Equations of state, reduced pressure $\beta P v_p$ vs volume fraction η, for ice cream cones.

The equation of state (figure 21, see page 50 for individual plots) looks similar to that of the single cones, except the nematic-crystal phase transition occurs at a much higher pressure. This seems to indicate that the spherical cap seems to hinder the formation of a crystal phase. The equations of state do not follow the equation of state of the infinite cones as closely as before, but they still have the same general shape.

The nematic order parameters also show a nematic-isotropic phase transition at more or less the same density as before, however for lower aspect ratios the jump is less sharp than before. The corresponding jump in density is also absent (or less prominent) in the equations of state, indicating that the phase transition may have become second order, at least for aspect ratios below some 'critical' aspect ratio. The effect is most pronounced for $L/D = 4$, where the nematic order parameter changes very gradually. Interestingly, this is also the region where the gyroid phase [3] was observed for tapered cones.

Unfortunately the corresponding snapshots (figure 23) do not show any long range order, but they do show that the ice cream cones start to align locally before the symmetry is broken and they align globally.

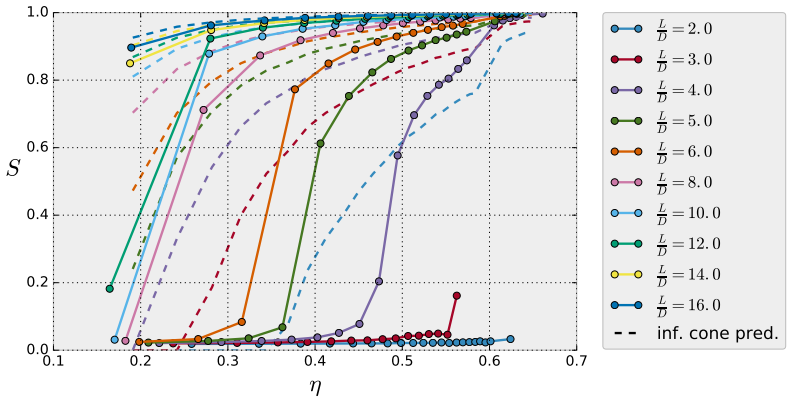
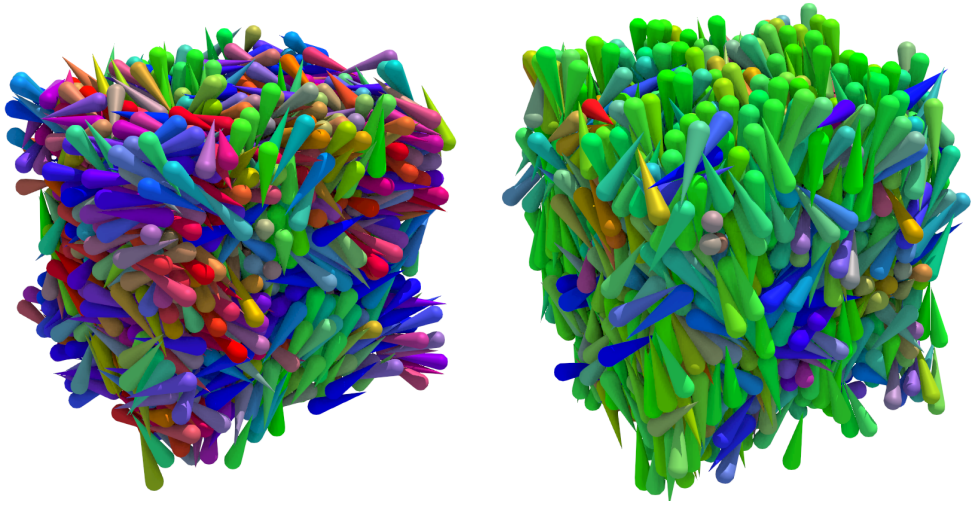


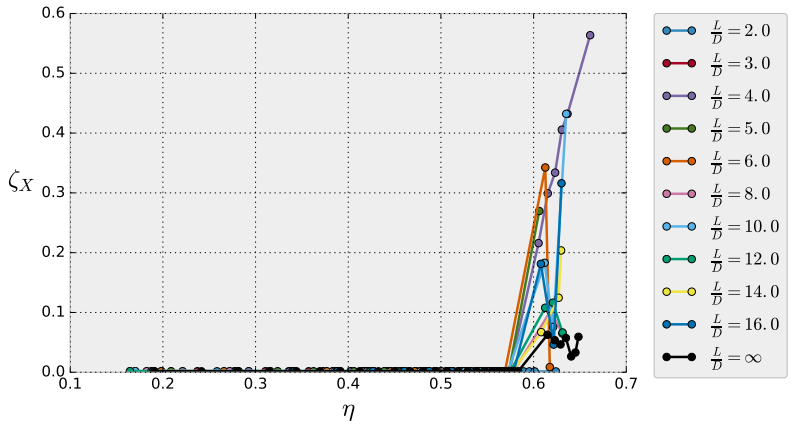
Figure 22: Nematic order parameter vs density of the ice cream cones. The dashed lines are predictions based on the value of $\mathrm{Var}(\theta)$ of the infinite cones.



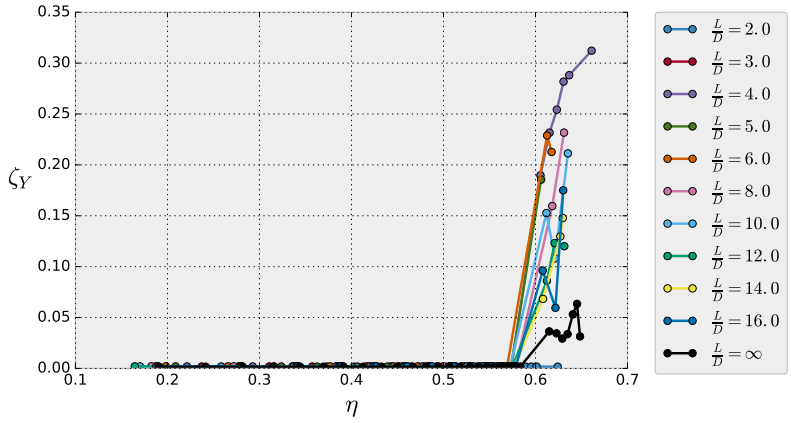
(a) Snapshot of $L/D = 4$ ice cream cones at $\beta P v_p = 8$

(b) Snapshot of $L/D = 4$ ice cream cones at $\beta P v_p = 9$

Figure 23: Snapshots of ice cream cones with $L/D = 4$ after $5 \cdot 10^6$ Monte Carlo steps.

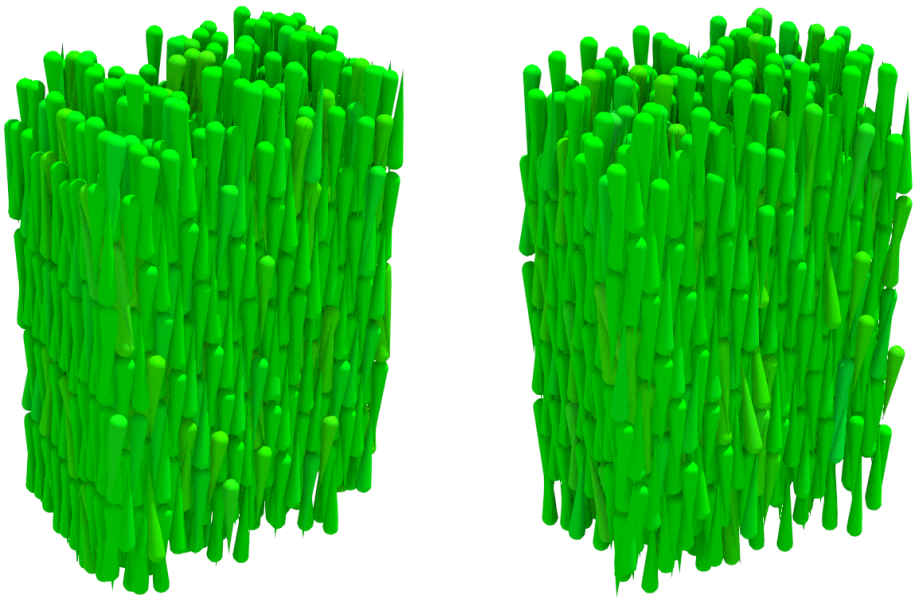


(a) Crystal order parameter along the x-axis ζ_X vs density for the ice cream cones.



(b) Crystal order parameter along the y-axis ζ_Y vs density for the ice cream cones.

Figure 24: Crystal order parameters



(a) Snapshot of ice cream cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 14$

(b) Snapshot of ice cream cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 15$

Figure 25: Snapshots of crystal configurations of ice cream cones with $L/D = 6$.

Since the crystal phase only forms at very large pressures, it is somewhat difficult to confirm whether the ice cream cones *also* prefer another crystal phase. At large pressures it becomes more difficult for the system to expand enough to restructure into a different crystal phase, which makes it more likely for the system to get stuck in a local optimum.

The ζ_X order parameter (see figure 24a on the preceding page) does show a sharp drop for $L/D = 6$, similar to the cones with $L/D = 3$, so perhaps those simulations could tell us something about a possible different crystal phase. Looking at the corresponding snapshots (figure 25 on the facing page) it does seem that the crystal becomes different, but the new crystal phase does not have a particularly clear structure, possibly because the large pressure is keeping it in a sub-optimal state.

The phase diagram (see figure 26) looks very similar to the one of the single cones, although because of the very high pressure required to form a crystal phase at lower aspect ratios, the exact position of the nematic-crystal phase transition is somewhat uncertain. Similarly the exact position of the isotropic-nematic phase transition for the higher aspect ratios is still unknown.

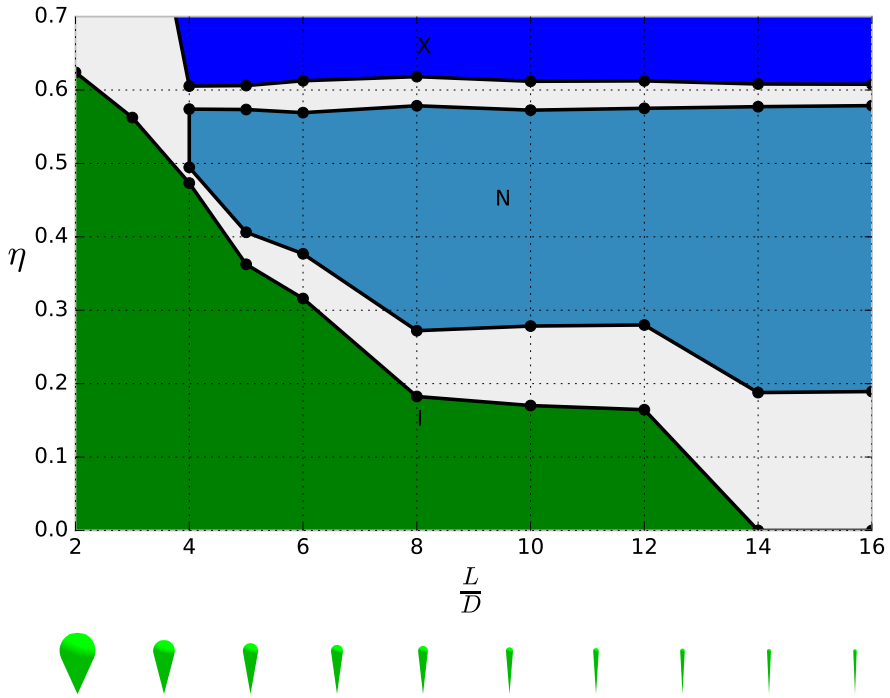
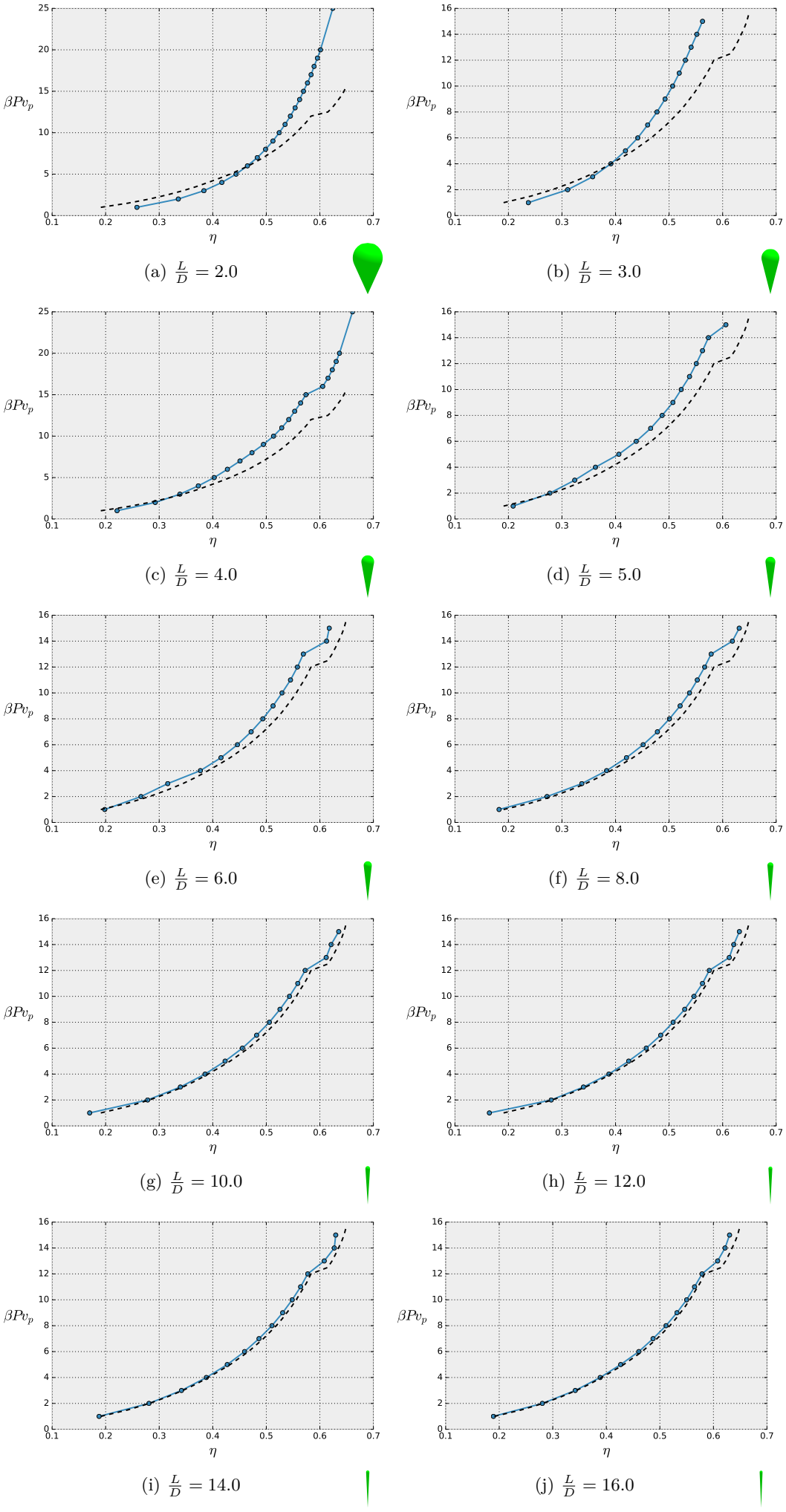


Figure 26: Phase diagram of the ice cream cones.



(a) $\frac{L}{D} = 2.0$ (b) $\frac{L}{D} = 3.0$

(c) $\frac{L}{D} = 4.0$ (d) $\frac{L}{D} = 5.0$

(e) $\frac{L}{D} = 6.0$ (f) $\frac{L}{D} = 8.0$

(g) $\frac{L}{D} = 10.0$ (h) $\frac{L}{D} = 12.0$

(i) $\frac{L}{D} = 14.0$ (j) $\frac{L}{D} = 16.0$

Figure 27: Individual equations of state for ice cream cones, with the infinite cones for comparison

5.3 Double Cones

Moving on to the double cones, again simulations were run for roughly 2000 particles for $5 \cdot 10^6$ Monte-Carlo steps. Looking at the equation of state (figure 28, see figure 33 on page 57 for individual plots), shows a clear phase transition to the crystal phase, which occurs at a relatively low pressure.

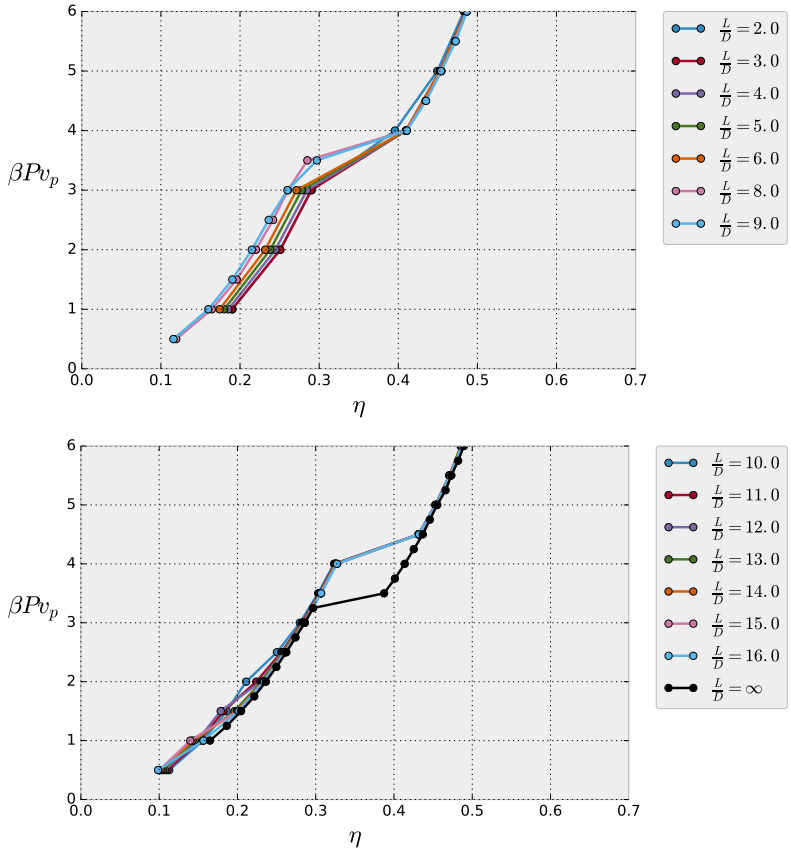
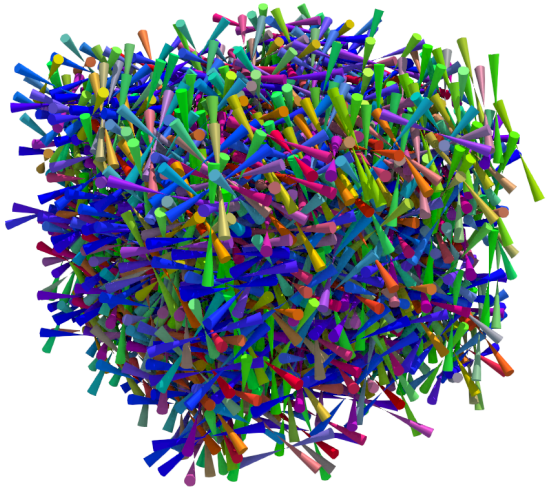
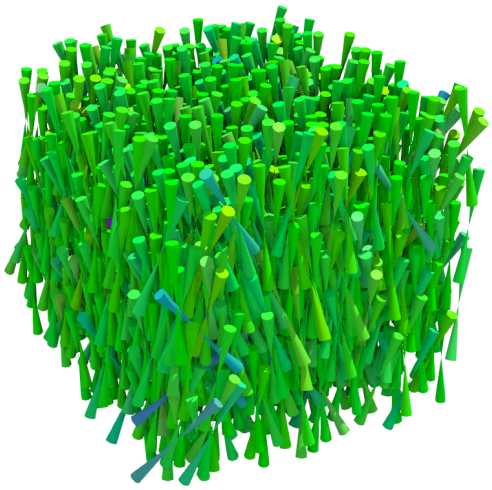


Figure 28: Equations of state, reduced pressure $\beta P v_0$ vs. volume fraction η, for the double cones.

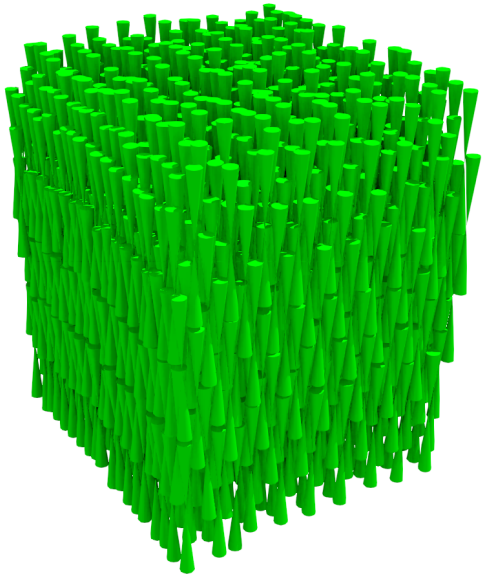
Snapshots of the configuration after $5 \cdot 10^6$ steps (figure 29 on the following page), again show a clear isotropic, nematic, and crystal configuration. The crystal configuration also seemed to be the same as the one used for the initial condition, no evidence of any different crystals was found. With the low pressure at which the crystal forms it is also not particularly likely that it is simply stuck in a local optimum, although the shape of the double cone might make it more difficult for cones to reorder themselves.



(a) Snapshot of double cones at $\beta P v_p = 1$



(b) Snapshot of double cones at $\beta P v_p = 3$



(c) Snapshot of double cones at $\beta P v_p = 5$

Figure 29: Snapshots after $5 \cdot 10^6$ Monte Carlo steps of double cones with $L/D = 10$ at different pressures.

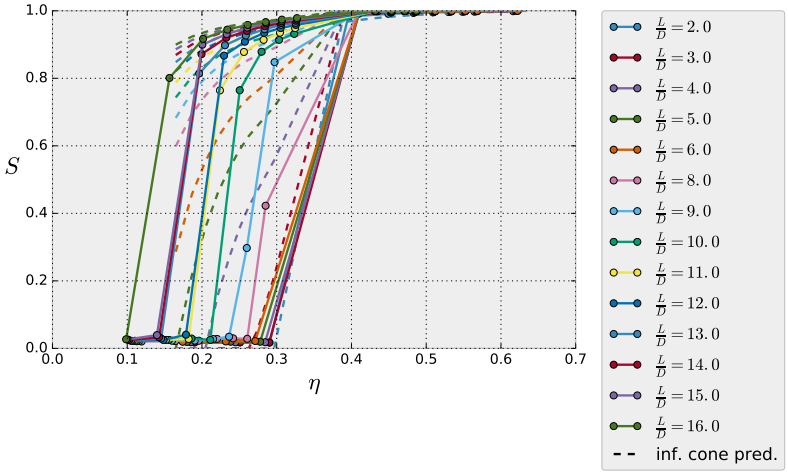


Figure 30: Nematic order parameter vs density of the double cones. The dashed lines are predictions based on the value of $\mathrm{Var}(\theta)$ of the double infinite cones.

However since both the nematic order parameter (figure 30), and the crystal order parameters (figure 32 on page 55) show a clear jump at the corresponding phase transitions, it seems reasonably safe to conclude that the double cones are indeed not attempting to form into another crystal phase, and that for the double cones the initial condition is the most stable crystal. The jump in the periodicity along the z-axis (figure 32c on page 55) is a bit weak but it increases steadily, the low initial value is likely because of small local fluctuations of the particle positions along the z-axis.

The location of the isotropic-nematic phase transition is more or less the same as for the single cones, however the isotropic-crystal phase transition occurs much earlier, which seems to be why double cones do not have a nematic phase until an aspect ratio $L/D \geq 9$. Therefore the resulting phase diagram (figure 31 on the following page) shows a very large crystal phase.

This early isotropic-crystal phase transition, seems to also prevent them from forming any other interesting phases. So, even though intuitively one might think that they should start to twist when forced together, in practice they will rearrange to form a crystal in a way that allows them to fit together without twisting.

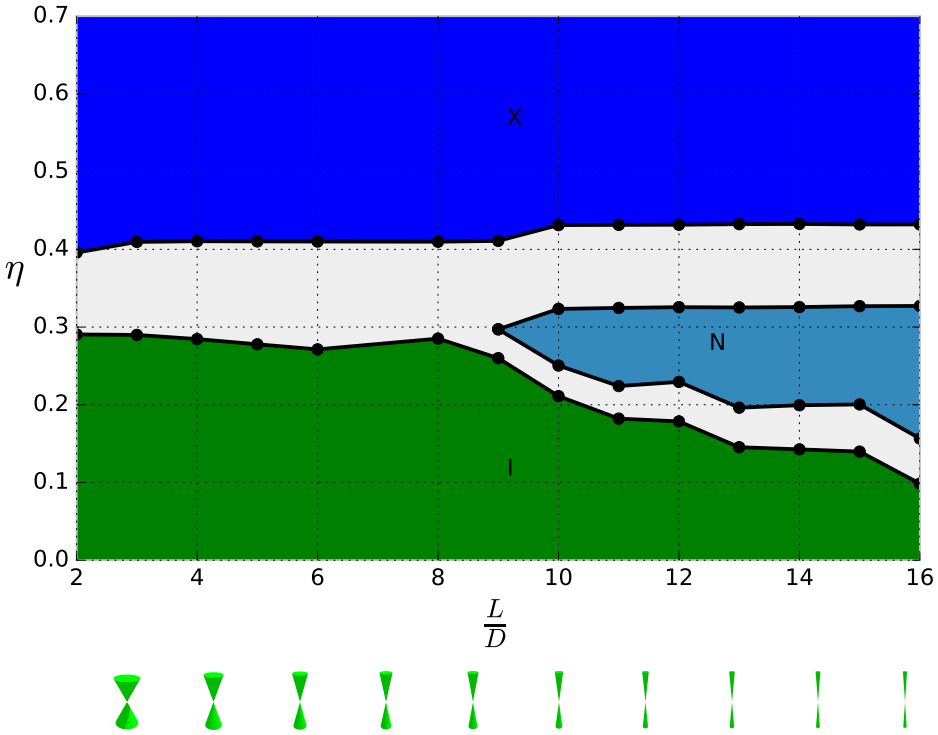
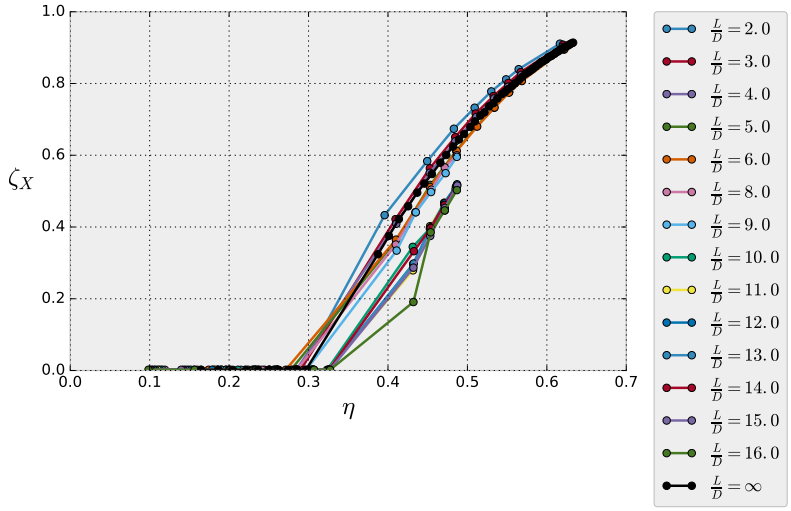
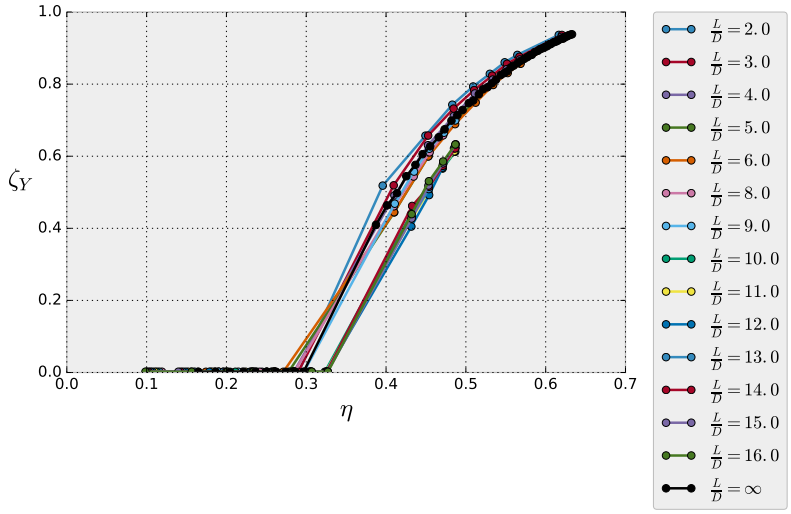


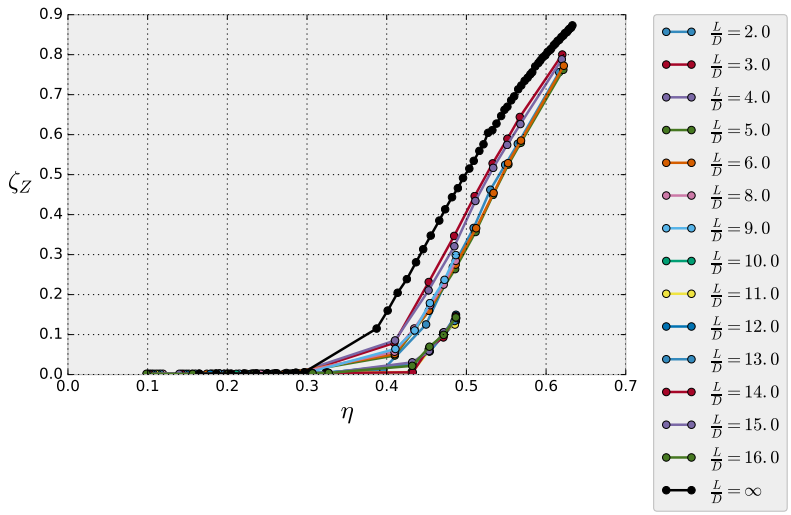
Figure 31: Phase diagram of the double cones.



(a) Crystal order parameter along the x-axis ζ_X vs density for the double cones.



(b) Crystal order parameter along the y-axis ζ_Y vs density for the double cones.



(c) Crystal order parameter along the y-axis ζ_Z vs density for the double cones.

Figure 32: Crystal order parameters

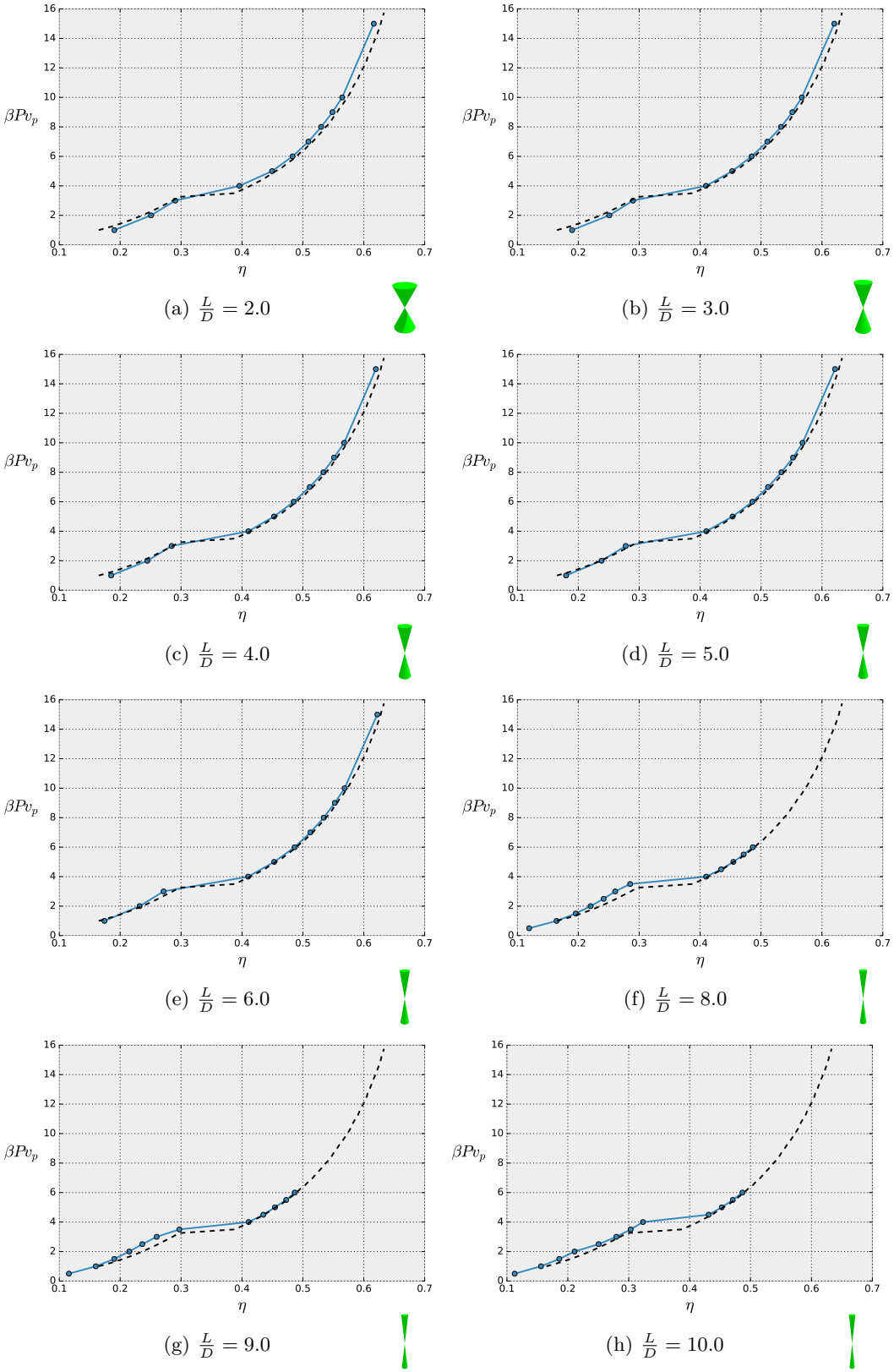
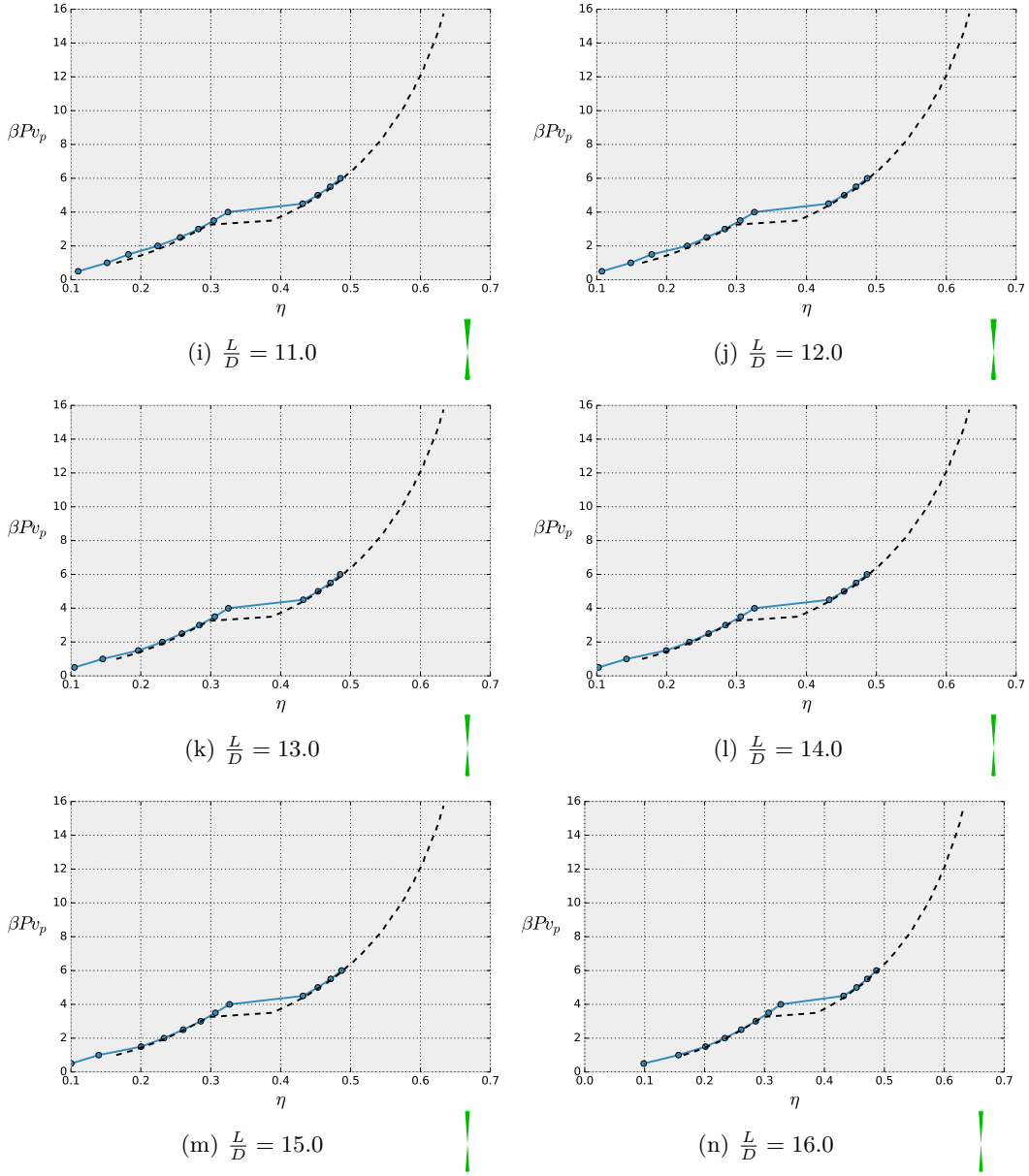


Figure 33: Individual equations of state for the double cones, with the double infinite cones for comparison



(i) $\frac{L}{D} = 11.0$ (j) $\frac{L}{D} = 12.0$

(k) $\frac{L}{D} = 13.0$ (l) $\frac{L}{D} = 14.0$

(m) $\frac{L}{D} = 15.0$ (n) $\frac{L}{D} = 16.0$

Figure 33: Individual equations of state for the double cones, with the double infinite cones for comparison

5.4 Double Ice Cream Cones

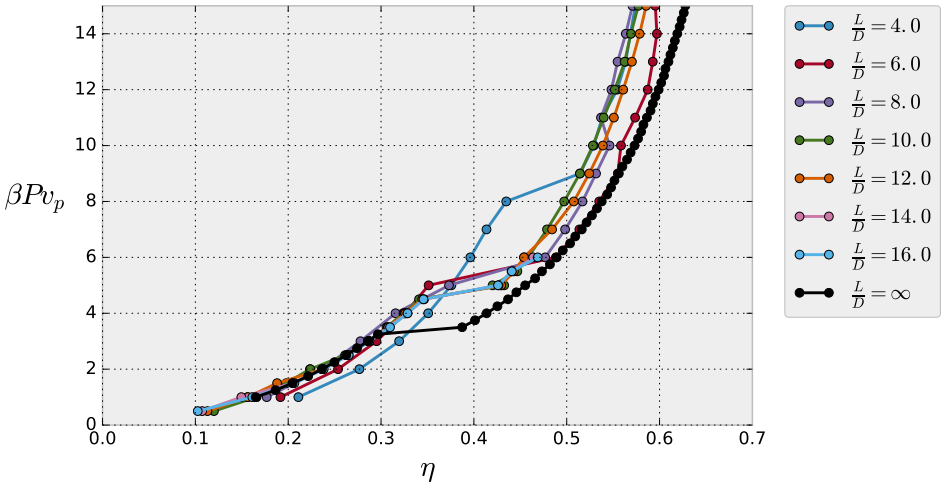


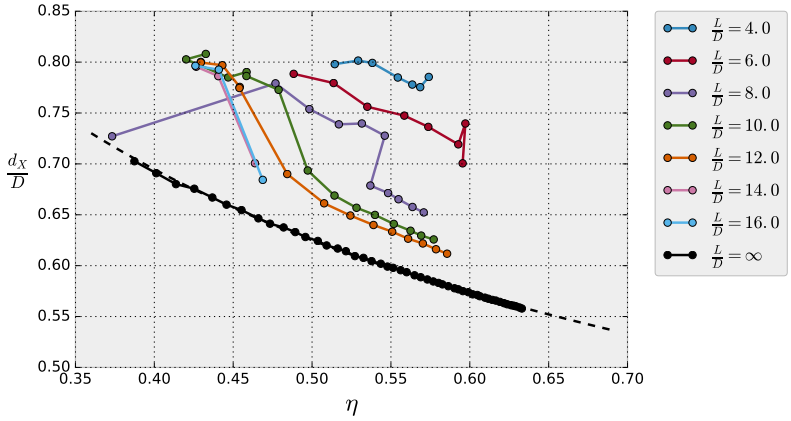
Figure 34: Equation of state for the double ice cream cones.

Since the ice cream cones seem somewhat more reluctant to form crystals, and the double cones form crystals very easily it is interesting to see what kind of behaviour the double ice cream cones end up with. The equation of state (figure 34, see figure 39 on page 62 for individual plots) do show some odd behaviour towards the higher pressures. In some extreme cases the density even decreases at higher pressures, which is not physically possible. This suggests that at high pressures the system is somehow stuck in a meta-stable state. This might indicate that, again, the initial condition was not the most favourable.

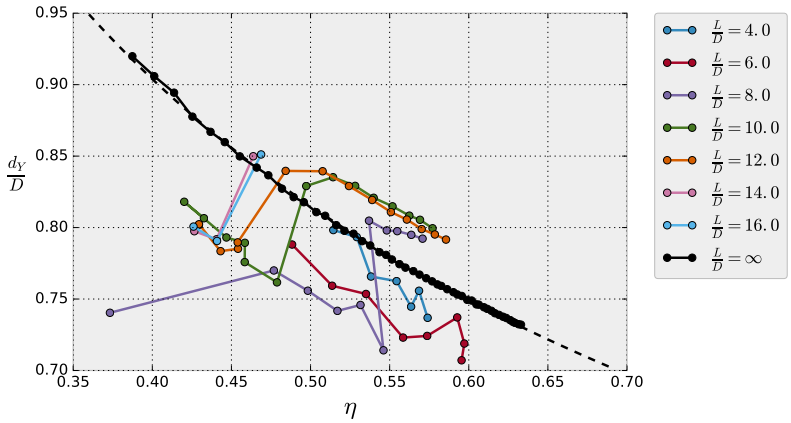
Some evidence for this can be found by looking at the structure of the crystal. The crystal order parameter not only measures the periodicity, but also implicitly measures the period with greatest periodicity. Measuring this period in units of D along the x and y axes results in figure 35.

For most aspect ratios there is a clear jump where at higher densities the period becomes closer to that of the infinite cones, but not at lower densities. Altogether this suggests that the initial condition is meta-stable (at least for low pressures) and there is different stable crystal phase. This is also confirmed by looking at snapshots of the double ice cream cones of aspect ratio $L/D = 10$, taken after $5 \cdot 10^6$ steps (figure 36 on page 60). At high pressures the configuration is similar to the initial condition, but at low pressures the individual ice cream cones start to form layers, with each double ice cream spanning across 2 different layers. This also means the limit for layered packing of cones derived in the section on single cones applies, hence this crystal cannot exist for packing fraction significantly higher than $\frac{2}{3}\frac{\pi}{2\sqrt{3}} \approx 60\%$. Which suggest that there will be a phase transition somewhere before that point towards a denser crystal configuration.

It can also be seen that (as expected) the crystal expands when the density is lowered. It might be expected that the crystal expands in all directions equally, making the periods proportional to $\eta^{-1/3}$, but the best fitting power laws for the period of the infinite cones along the x and y axes are $0.505\left(\frac{\eta}{\pi/4}\right)^{-0.473}$ and $0.661\left(\frac{\eta}{\pi/4}\right)^{-0.464}$ respectively. It appears

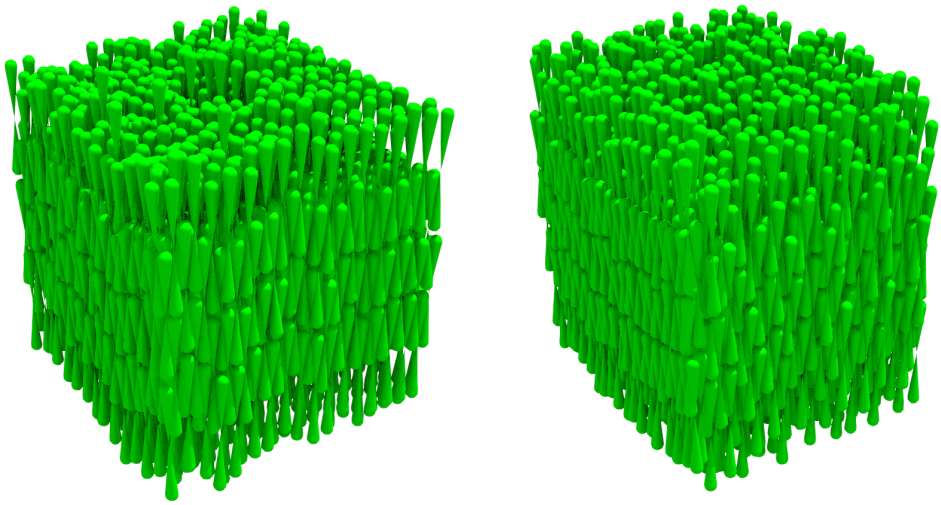


(a) Period along x axis in units of D, versus the density. The dashed line corresponds to $0.505\left(\frac{\eta}{\pi/4}\right)^{-0.473}$.



(b) Period along y axis in units of D, versus the density. The dashed line corresponds to $0.661\left(\frac{\eta}{\pi/4}\right)^{-0.464}$.

Figure 35: Period along x and y axes in units of D, versus the density.



(a) Snapshot of double ice cream cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 6$

(b) Snapshot of double ice cream cones after $5 \cdot 10^6$ Monte Carlo steps at $\beta P v_p = 10$

Figure 36: Snapshots of crystal configurations of double ice cream cones with $L/D = 10$.

the period is approximately proportional to $\eta^{-1/2}$, suggesting that the crystal expands mostly along the x and y-axes, and not the z-axis. Therefore, to examine the structure independently from the density, the periods should be multiplied by something proportional to $\sqrt{\eta}$. Since the maximal density of the initial configuration is $\pi/4$ it makes most sense to multiply by $\sqrt{\frac{\eta}{\pi/4}}$. Corrected in this way the 'corrected' period should remain stable, and can be compared directly to the period of the initial configuration. This results in figure 37.

This figure clearly shows two different clusters, one of them close to the initial condition $(\frac{1}{2}, \frac{2}{3})$ but another close to $(0.6, 0.6)$, which presumably corresponds to the other crystal phase which seems to have the same structure along the x and y axes.

Unfortunately without more detailed knowledge on the new crystal phase it is not really feasible to find where the phase transition is between both crystal phases, however it is still possible to find the location of the isotropic-nematic and nematic-crystal phase transitions to construct a phase diagram (figure 38). Overall the phase diagram has not changed that much from the one for the double cones, the main difference is that the isotropic-crystal and isotropic-nematic phase transition occur at a somewhat higher density.

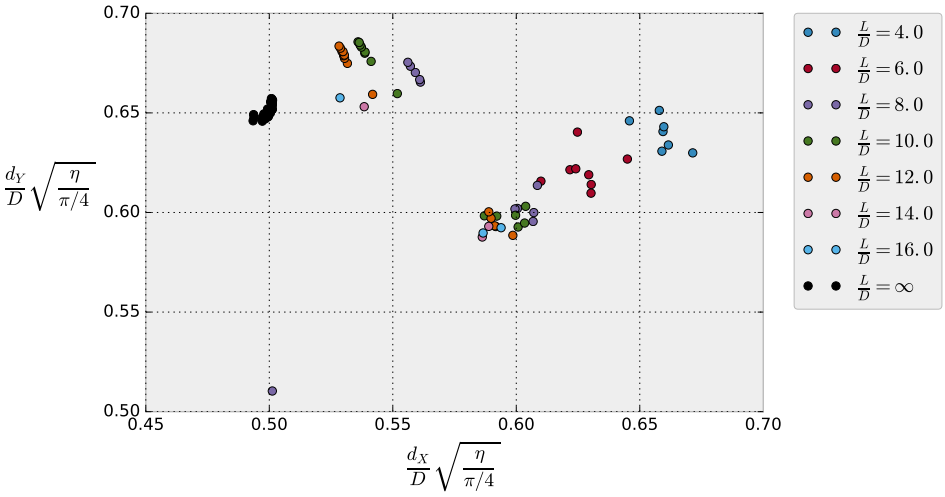


Figure 37: Period along x- and y-axes.

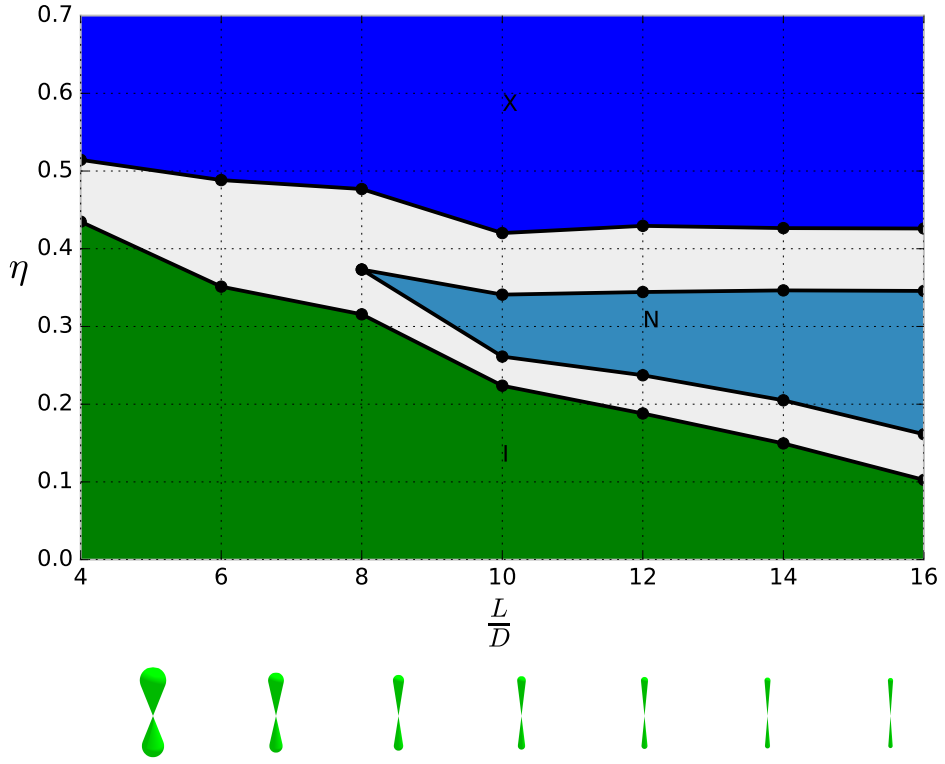


Figure 38: Phase diagram of the double ice cream cones.

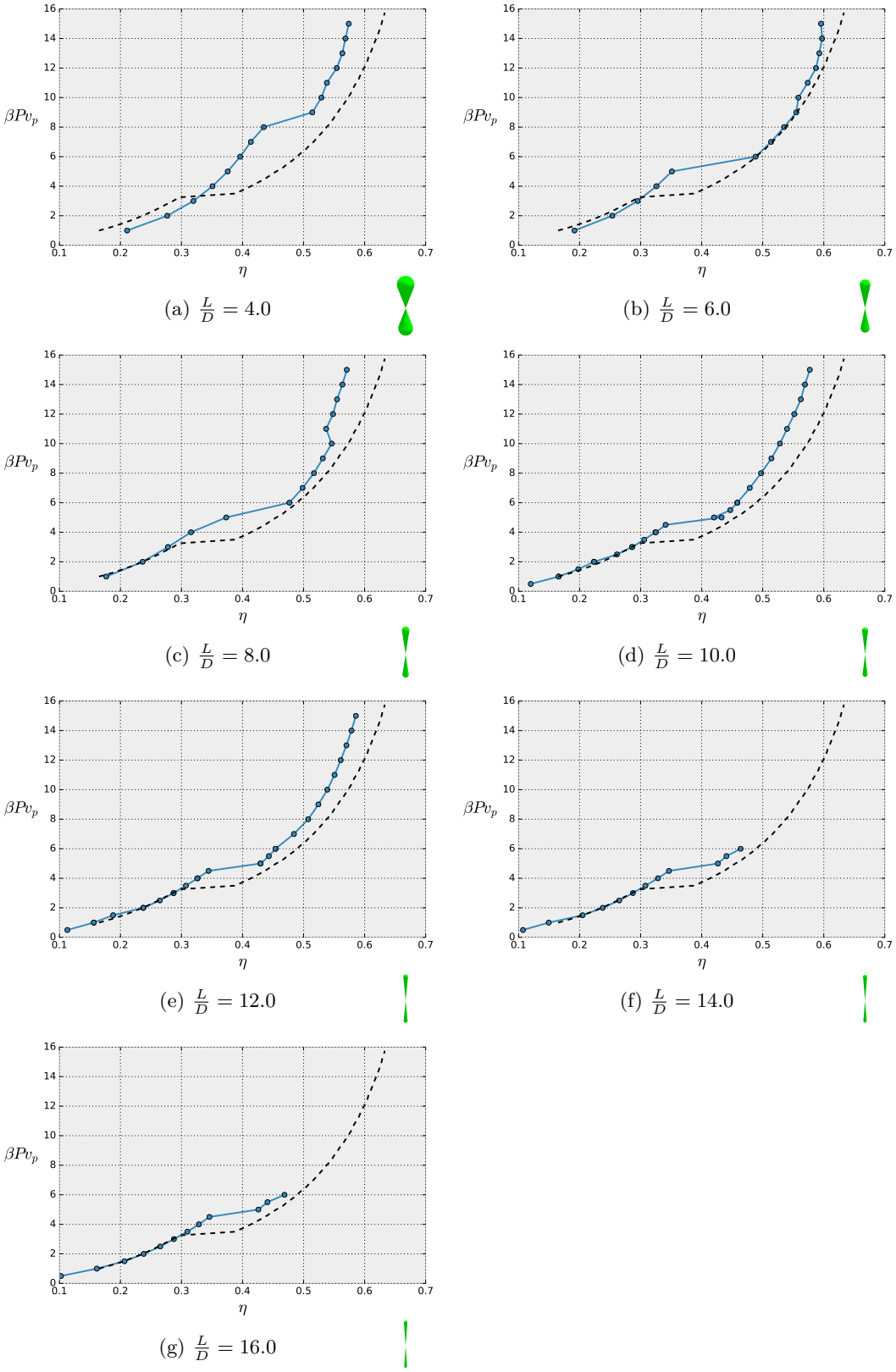


Figure 39: Individual equations of state for the double ice cream cones, with the double infinite cones for comparison

6 Conclusion

Cones do not form a smectic phase, regardless of aspect ratio, or whether the base is flat or spherical. The most likely explanation is that the sharp tip prevents the formation of the smectic phase. There are several reasons that this is likely to be the case. The first is that pear-like particles do seem to exhibit a smectic phase [14]. The second lies in the fact that by blunting the tip it is possible to transform a cone into a spherocylinder, this will be discussed in more detail in the following section.

Double cones also do not form a smectic phase, but only because they form a crystal rather easily, even at a packing fraction as low as 40%. This prevents the smectic phase from forming, and the nematic phase only forms for aspect ratios greater than 8. The double ice cream cones show similar behaviour, except they seem to form a different crystal at lower densities.

6.1 Outlook

If, as suspected, the sharp tip of the cones prevents a smectic phase from forming then this could also be of use in examining phases like the biaxial nematic phase, since for board like particles the smectic phase seems to suppress the formation of such a phase except for very elongated particles [15]. By stretching the cone in one direction (perpendicular to its axis) it can be made biaxial, and if there is no smectic phase then this could lead to a rather large biaxial nematic phase.

For future research the ice cream cone overlap detection could also be adapted to simulate other shapes. Such as the cones with a tapered tip (just by changing the range of the parameters) depicted in figure 40, which could be used to find how sharp the tip needs to be to prevent the smectic phase. Note that when the ice cream cones are blunt enough they *become* spherocylinders, so they will definitely exhibit a smectic phase at some point.

Figure 40: Interpolation from ice cream cone to spherocylinder.

Instead of an ice cream cone, which can be considered a union of spheres with a linear relationship between the position of the spheres and their radii, the overlap detection techniques can easily be generalized to other relationships. In particular something like a 'spherohyperboloid' (i. e. a hyperboloid with two spherical caps) could be simulated using a radius of $\sqrt{a + bt^2}$ (where the location of the centre of the sphere still depends linearly on t). Note that a constant radius corresponds to a spherocylinder, and a linear radius to a (double) ice cream cone, hence both can be considered degenerate cases of the spheroyhyperboloid (suggesting that 'spherocone', might have been a better name than ice cream cone). This makes the spherohyperboloid an interesting candidate for examining the relation between (double) ice cream cones and spherocylinders.

The double cones are also interesting because of their incredibly large crystal phase, occurring at packing fractions as low as 40%. With an upper limit of around $\pi/4 \approx 79\%$ this implies the crystal can expand almost 2 times in size before it melts, which could have

interesting applications if it can be realized experimentally.

Unfortunately, because of this crystal phase, no evidence of 'twisting' was found for the double cones but using spherohyperboloids to interpolate from spherocylinders to double cones such 'twisting' might affect the smectic phase, which should either disappear or transform in a new 'twisted' phase.

The techniques used for the infinite cones are themselves a generalization of a technique which was applied to spherocylinders. Hence they readily generalize to other shapes. Similarly the techniques could also be used to go the other way and examine the $L/D = 0$ limit where cones become more like thin discs (yet not the same). More generally the technique suggests that a symmetry, or even approximate symmetry of a configuration space across different shapes can provide a lot of information of that particular family of shapes (it is interesting to note that the configuration space of 'infinitely elongated' is not even capable of describing particles *not* aligned to the nematic director). The definition of the 'infinite' aspect ratio limit also looks similar to the mathematical definition of a direct limit, which hints at a more mathematical description of various 'physical' limits (e. g. low density limit, infinite pressure limit).

References

[1] Lars Onsager. The effects of shape on the interaction of colloidal particles. *Annals of the New York Academy of Sciences*, 51(4):627–659, 1949.

[2] Peter Bolhuis and Daan Frenkel. Tracing the phase boundaries of hard spherocylinders. *The Journal of Chemical Physics*, 106(2):666–687, 1997.

[3] L. J. Ellison, D. J. Michel, F. Barmes, and D. J. Cleaver. Entropy-driven formation of the gyroid cubic phase. *Phys. Rev. Lett.*, 97:237801, 2006. doi: 10.1103/PhysRevLett.97.237801.

[4] F Hagemans, E. B. van der Wee, A. van Blaaderen, and A. Imhof. Synthesis of cone-shaped colloids from rod-like silica colloids with a gradient in the etching rate. *Langmuir*, 32(16):3970–3976, 2016. doi: 10.1021/acs.langmuir.6b00678. PMID: 27046046.

[5] P.G. Gennes and J Prost. *The physics of liquid crystals*. Number 83. Oxford university press, 1995.

[6] 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.

[7] MEJ Newman and GT Barkema. *Monte Carlo Methods in Statistical Physics chapter 1-4*. Oxford University Press: New York, USA, 1999.

[8] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988. ISSN 0882-4967. doi: 10.1109/56.2083.

[9] MP Allen and DJ Tildesley. Computer simulation of liquids oxford university press oxford 385. 1987.

[10] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[11] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[12] Gino van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999. doi: 10.1080/10867651.1999.10487502.

[13] J.R. Banavar A. Maritan A. Trovato, T.X. Hoang. Symmetry, shape, and order. 2007. ISSN 0027-8424. doi: 10.1073/pnas.0707523104.

[14] Frédéric Barmes, M Ricci, Claudio Zannoni, and DJ Cleaver. Computer simulations of hard pear-shaped particles. *Physical Review E*, 68(2):021708, 2003.

[15] Stavros D Peroukidis and Alexandros G Vanakaras. Phase diagram of hard board-like colloids from computer simulations. *Soft Matter*, 9(31):7419–7423, 2013.

[16] S Dussi. When shape is enough: from colloidal spheres to twisted polyhedra, from icosahedral to chiral order. 2016.

[17] Nick Tasios. partviewer3d-glsl, an opengl based visualization program for 3d particles. `https://github.com/Grieverheart/partViewer3D-GLSL`, 2016.

Acknowledgements

I would like to thank my supervisors (Simone Dussi, Marjolein Dijkstra, Nick Tasios) who helped shape this thesis with their comments and suggestions during our (mostly weekly) discussions. I would specifically like to thank Simone Dussi for sharing information from his own (by now finished, and well received) research [16] which provided much of the inspiration of this thesis, Nick Tasios for his excellent visualization program [17] (responsible for the snapshots in this thesis), and Marjolein Dijkstra for helping me find this subject. Also thanks to Rob Bisseling for helping me combine mathematics and physics, and René van Roij for helping me set up this project. Finally I would also like to thank the other master students of the Soft Condensed Matter Group for the occasional inspiration and making the student room 'gezellig'.