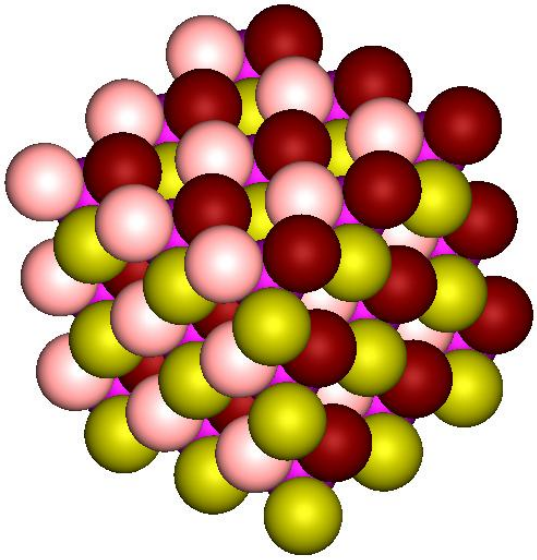
Monte Carlo simulations of tetrahedral clusters of spheres



Gaia Steinbuch

Supervisor: Prof. dr. M Dijkstra
Soft Condensed Matter
Debye Institute for Nanomaterials Science
Faculty of Science
Utrecht University
Student no.: 0481904

Universiteit Utrecht

Summary

In this work we investigate colloidal systems with tetrahedral clusters. The clusters are modelled as hard spheres. Previously, work has already been done using experiments and simulations. It is possible to make these clusters using the emulsion droplet evaporation method. What we were interested in is seeing how the particles would behave in simulations. The theory of molecular Monte Carlo simulations and the techniques used are explained. The molecular simulations show that the tetrahedral clusters behave as would be expected: at high pressures they are a solid (we put them in the fcc structure) and at low pressures they form a liquid.

Preface

This project has been done to obtain my master's degree in Nanomaterials: Chemistry & Physics. This project has been very educational for me. I've learned a lot about doing research, simulations, (C-)programming, using Linux, plotting, writing, presenting, and many other things. First, I would like to thank my supervisor professor Marjolein Dijkstra in particular, for taking the time to talk with me on a nearly weekly basis (always interesting and often with new tips of things to try) and for allowing me to do this project at a slower pace than nominally, part of which at home. Also I would like to thank Jos Koeckhoven, for all his care and help he has given me. In addition, I would like to thank the group Soft Condensed Matter for their support and interesting work discussions, as well as my fellow students with whom I shared a room for the interesting times. I thank Jeroen van der Schot in particular, for his help with Linux and my programming. Lastly I would like to thank my family and Michael. Your support has been invaluable to me. You were very understanding and always listened to my stories and problems. You inspired me.

List of symbols

Symbol	Meaning
A	A certain (observable) property
A-D	The four particles of a tetramer
acc	Probability of accepting a trial move
c	Chosen constant
C	The number of contact spheres
d	Dimensionality of the system
E	Energy
E_i	The energy of a state $\|i>$
$f(x)$	A function
F	Helmholtz free energy
h^{dN}	Volume in classical space
$\hbar$	Planck's constant
$\mathcal{H}$	Hamiltonian of a system
I	A one-dimensional integral
i	A counter
i	Imaginary unit
j	Another counter
j	Imaginary unit
k	Imaginary unit
k_b	Boltzmann constant
K	Rotation matrix
$\mathcal{K}$	Kinetic part of the Hamiltonian $\mathcal{H}$
l	The previous configuration
L	A large number
m	Points in a configuration
M	The mass of the particles
M_2	Second moment of mass
n	A new configuration
N	Amount
$\mathcal{N}$	Probability density
o	An old configuration
P	The pressure
p_{1x}	The momentum of the first particle in the x-direction
P_i	Probability to find a system in state i
p^N	Momenta of the particles
q	A four-dimensional vector
Q	Partition function
r^N	Positions of the particles
r_0	The centre of mass of a cluster
r_i	The position of the centre of the particle
R	Radius of a sphere

$realpart$	The amount of particles that are inside the simulation box
$rn2$	A random number in the interval $[0,1]$
S	Entropy
t	Time
T	Temperature
T	A matrix describing the coordinates of the particles in a fixed position
u	The amount of units in a simulation, the amount of clusters plus big spheres
U	A coordinate system, described by a matrix
$\mathcal{U}$	Potential part of the Hamiltonian $\mathcal{H}$
U_1	The first vector in the coordinate system U
U_m	The depth of the pair potential
$units$	The amount of units in the box (big spheres + tetramers)
V	The (average) volume
V_{new}	The new volume
V_{old}	The old volume
$w(x)$	Non-negative probability density
x	A variable, a vector, also x-coordinate
x_1	The x-coordinate of the first particle
x_{U_1}	The x-coordinate of the vector U_1
Z	Configurational part of the partition function

α	Probability of performing a trial move
α_s	Size ratio of large to small sphere radii
α_c	Critical size ratio of large to small sphere radii
β	$1/T$
γ_{dl}	The surface tension of the droplet-liquid interface
γ_{pl}	The surface tension of the particle-liquid interface
γ_{pd}	The surface tension of the particle-droplet interface
η	The packing fraction
λ	A variable
π	Transition probability
ρ	The density
σ	Diameter of a sphere
σ_I^2	Variance
Ψ	Wavefunction of the system
Ω	A number of eigenstates

Contents

Chapter 1. Introduction

An important research area in material science is the prediction of macroscopic properties of fluids and solids using computer simulations. In this work, we focus on colloidal clusters of spheres (tetrahedral clusters) and investigate the phase behaviour when the pressure is changed. We also wanted to investigate tetrahedral clusters mixed with bigger spheres and whether, or not they will form a $MgCu_2$ structure (see Fig. 1.1d, 1.1e, 1.1f and Figure 1.2). This structure would allow the crystal to have a photonic bandgap in the visible region which would be useful for applications.

We study the phase behaviour of tetrahedral hard spheres (tetrahedral clusters) mixed with single hard spheres of a certain diameter (we call this the binary system), using Monte Carlo simulations, and we investigate whether the particles will form this structure. A tetrahedral cluster consists of four spheres in a tetrahedral arrangement. Certain colloidal particles can be modelled as hard spheres [1]. The reason could be that the colloidal particles are sterically stabilised, this may lead to a reduction of the Van der Waals-forces.

The problem formulation of this work is as follows.

> What will happen to tetrahedral clusters and binary systems under certain conditions? Under what condition and for which colloidal particles, will $MgCu_2$ structures be formed? Can Monte Carlo simulations provide a framework with which this can be analysed and predicted?

First we discuss some experiments and simulations done in literature, in Chapter 2. Then we will provide an introduction into Monte Carlo simulations, in Chapter 3, explaining the technique and specific details of the programming. In Chapter 4, the set up parameters of the simulations of the particles will be discussed, and in Chapter 5 the main results will be shown. Finally, the report will be finished in Chapter 6 with a summary of the main results, in the form of conclusions.

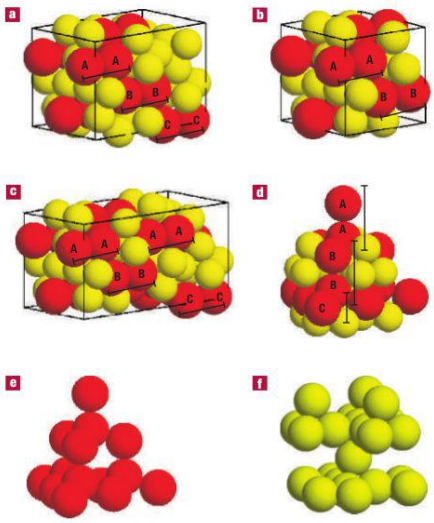


Figure 1.1: Two different types of particles forming different structures. Figure d is the $MgCu_2$ structure. Figures e and f are the individual structures the $MgCu_2$ structure is built of. [2]

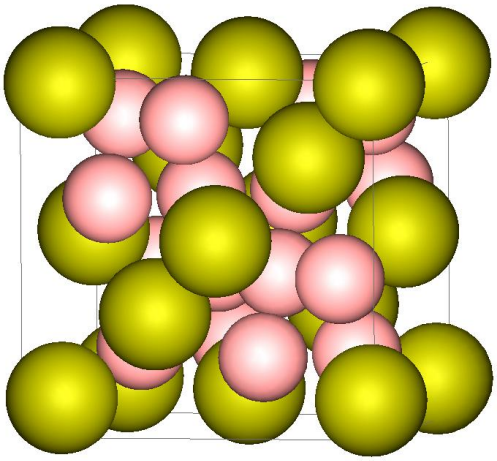


Figure 1.2: Two different types of particles forming a $MgCu_2$ structure.[3]

Chapter 2. Previous experimental and simulation work on tetrahedral clusters

In this chapter we examine the methods available experimentally to make tetrahedral clusters. We also examine some theoretical work (simulations).

The reason many people are interested in tetrahedral clusters, is because making 3D photonic crystals will be a lot easier if the colloids form crystals with a diamond lattice. Tetrahedral clusters might do that [4].

The first people who made tetrahedral clusters experimentally, are Manoharan et al. [5]. In 2003, they proposed the emulsion droplet evaporation method. Emulsion droplets with particles are suspended in a liquid, after which the solvent is evaporated. This causes the size of the droplets to reduce and allows the particles to slowly form a densely packed cluster. For cluster sizes up to $n = 16$ it has been shown that this method leads to well-defined cluster shapes for a variety of particles and solvents.

The particles used by Manoharan et al. [5] are crosslinked polystyrene microspheres, 844 nm in diameter (2.5% polydispersity). They have sulfate groups covalently bounded to the surface. In pure water the sulfate groups dissociate, making the surfaces of the spheres charged and preventing van der Waals interactions to aggregate the spheres. In an organic solvent, such as toluene (a good solvent for polystyrene), the sulfate groups do not dissociate much, but the van der Waals forces are much smaller than in water. The particles swell up with solvent and interact only through a short-ranged steric repulsion. This makes that they can be approximated as hard spheres.

Manoharan et al. use a system that contains both water and toluene. The spheres are dispersed in toluene, water is added and the whole system is mixed to create an oil-in-water emulsion consisting of small droplets of toluene ranging from 1 to 10 μm in diameter. The particles are bound to the droplet interfaces by surface tension. The toluene is then evaporated, forcing the particles in each droplet to densely pack together. The critical point of this evaporation process is a stable intermediate state called a spherical packing, formed when the particles touch each other on the surface of the droplet. Removing more toluene at this point causes the droplet to deform. This will generate capillary forces that lead to a fast rearrangement of the particles ($< 33\ ms$). When the toluene evaporates, the particles deswell. At that point the interparticle van der Waals interactions increase and the particles stick to each other, forming a small colloidal aggregate. Thus due to capillary forces the particles eventually form their final configuration; while they act as hard spheres, the van der Waals forces cement the spheres together. A side effect of this method is that the surfaces of the particles on the outside of the cluster are exposed to water earlier than the surfaces on the inside. The disassociation of surface charges on the outer surfaces prevents the cluster from aggregating with other clusters.

After the toluene has been evaporated, one is left with clusters of various sizes, because the initial droplets are not equal in size. By centrifuging them in a density gradient the clusters can be separated. Clusters of a given n are all identical.

Manoharan et al. also prepared clusters based on 900 nm silica spheres and 1170 nm polystyrene spheres [5] [6].

It was suggested by Manoharan et al. [5] that the clusters are formed in two steps. The colloids first reach an optimal packing confined to the surface of a sphere and are then pulled towards the centre by capillary forces. The particles are restricted to a continuous and smooth surface. The second step minimises the second moment of the mass distribution of the cluster:

$$M_2 = \sum_{i=1}^{N} |r_i - r_0|^2 \qquad (2.1)$$

where r_0 is the mass of the cluster and r_i the position of the centre of the particle. This mimics the rearrangement of the cluster in the last stage of evaporation, where the shape of the emulsion droplet strongly depends on the particles [5][7].

By altering the salt concentration in the liquid and applying an electric field, the interactions between the particles can be tuned. This increases the range of possible colloidal structures [4].

Manoharan and Pine proposed in 2004 to first make building blocks in the form of colloidal clusters, and then use these building blocks to make crystals [8].

Yi et al. [9] developed a method that is capable of making many identical clusters, where the number of spheres n can be varied between 2 and 15. Their process is based on the emulsification of a suspension of lightly crosslinked polystryene microspheres in toluene (with an aqueous surfactant). This results in a toluene-in-water emulsion with the polystyene microspheres bound by surface tension to the droplet interfaces. When the toluene is evaporated, the particles form stable clusters. Yi et al. use spheres of slightly different materials than Manoharan et al. [5]. Yi et al. [9] showed the process for two kinds of spheres. The first are silica microspheres stabilised with octadecyl (C_{18}) chains on their surfaces. The second are non-crosslinked PMMA (poly(methylmethacrylate)) microspheres stabilised by PDMS (poly(dimethilsiloxane)) chains on their surfaces.

The polystyrene microspheres are lightly crosslinked. When they are placed in a good solvent (such as toluene), they swell but keep their spherical shape. In toluene, they are stabilised (against aggregation) by steric repulsion. The microspheres are made with ionisable surface groups. These groups dissociate in water, thus making the surfaces of the particles charged. This prevents aggregation also. These two properties make the spheres stable in both water and a non-polar solvent (oil).

Another difference with Manoharan et al. [5] is that Yi et al. [9] suspend either types of particles in the same solvent (to better compare results), namely a hexane, not toluene. The hexane

suspension is mixed with the aqueous surfactant solution. The surfactant is a triblock copolymer of poly(ethylene oxide) and poly(propylene oxide)(plurionic P123, BASF, consisting of EO_{20}-PO_{70}-EO_{20}, with EO being ethylene oxide and PO propylene oxide). The resulting solution contains hexane droplets with sterically stabilised silica or PMMA spheres attached to the droplet interfaces. The hexane water interfaces are stabilised by the P123 triblock copolymer. The hexane is slowly evaporated and the colloidal clusters then form. It is important the P123 surfactant has to stay in the solution, because it prevents aggregation between the clusters. The PMMA and silica clusters are stable in water, but the PMMA clusters do not maintain their shape when dried on a substrate. Moreover, the spheres can easily be rendered unstable by ie. diluting the solution. To avoid this problem Yi et al. added a thin layer of silica to permanently encapsulate the clusters. The resulting clusters are stable in water without surfactant and are charged [9].

In 2005 Cho et al. [10] proposed to make colloidal clusters of crosslinked polystyrene using water-in-oil emulsions (unlike Yi et al. [8], who made spheres in oil-in-water emulsions). This method is versatile since most of the commercially available colloids are suspended in aqueous media with much better stability. The spheres (230 nm in diameter) exhibit soft electrostatic repulsions. In a water-in-oil emulsions, the spheres will migrate to the centres of the aqueous droplets rather than sit at the oil-water interfaces. The polystyrene spheres were made by emulsifier-free emulsion polymerisation. Cho et al. also made silica colloidal clusters, using a similar method. The silica spheres were made using a modified Stöber method.

The method works as follows. The colloids are dispersed in toluene or mineral oil respectively (for polystyrene or silica spheres). Using seeded growth the polystyrene particles grow to 830 nm. The silica particles (in a similar experimental setup) are 800 nm in diameter. An aqueous polystyrene suspension is then added to the toluene and homogenised. The next step is to slowly evaporate the water. The colloidal clusters are formed during this step. Afterwards, the oil phase is removed and the clusters are redispersed in water so they can be seperated based on the amount of spheres in the clusters via density gradient ultracentrifugation. From this experiment several clusters are found (with a varying number of n) [10].

Zerrouki et al. [6] adjusted the method Manoharan et al. developed [5], in 2006. They prepared monodisperse emulsions [6], which allows them to obtain high yields. They made clusters of two, three or four silica particles with a radius of 1.2 μm.

First, the silica particles were synthesized. Next, the surface of the spheres was made hydrophobic by chemical grafting. The silica particles were dispersed in octane and mixed with an aqueous solution. Controlled shear was performed to obtain a quasimonodisperse emulsion. Removing the octane led to an aqueous suspension containing the colloidal clusters. Finally, the clusters were seperated by sedimenting them in a liquid density gradient column.

Zerrouki et al. [6] had a higher yield of doublets, triangular and tetrahedral clusters than Manoharan et al [5].

The composition of the cluster suspension is highly influenced by the percentage of silica, the

viscosity of the continuous phase and the shear rate. Zerrouki et al. varied the silica concentration from 1% to 40% (w/w) in the octane phase. If the contentration of silica is low (1%), mainly single particles are found. There are mainly doublets for concentrations between 1% and 5%, between 5% and 10% mainly triplets and tetrahedral clusters are most present when the silica concentration is between 10% and 17% (independent of the shear rate used). When the amount of silica is greater than 17%, single particles dominate again. Optical microscopy reveals the silica spheres may adsorb on the water-oil interface, thus changing the viscoelastric properties of the oil droplet interface. This could change the fragmentation mechanism.

The fraction of clusters containing more than four particles does not exceed 35% if the continuous phase is sufficiently viscous (30% of PVP in the continuous phase). This achieves effective fragmentation of the oil droplets that initally contain a high concentration of particles, which leads to a high percentage of small aggregates.

If one uses a low to moderate shear rate ($\gamma^{'} = 1458 - 4165\ s^{-1}$) there are a high number of small aggregates. At a high shear rate ($\gamma^{'} = 8831\ s^{-1}$) single particles are favoured because the high viscous stress leads to fragmentation of droplets containing only one particle.

The best results were obtained using low to moderate shear rates, a highly viscous continuous phase and intermediate amounts of silica (5%-17%) in the droplet phase [6].

Guangnan et al. [11] provided experimental measurements of hard spheres with a short-range interaction. There is a depletion interaction with a range of 1.05 times the particle diameter and a depth of about $4k_bT$ (where k_b is the Boltzmann constant and T the temperature). The pair-potential is short-ranged. The total potential energy U can be approximated by:

$$U = CU_m \qquad (2.2)$$

where C is the number of contacts and U_m the depth of the pair potential.

The clusters were created by isolating a small number of polystyrene microspheres in cylindrical microwells filled with water and poly(N-isopropylacrylamide)(polyNIPAM) nanoparticles (which cause the depletion interaction). The microwells were chemically altered so the particles would not stick to the surfaces. This allows the clusters to form in the middle of the wells. Once the clusters reached equilibrium, optical microscopy was used to observe the cluster structures. The number of particles per well is not controlled, however there are enough clusters with $n < 10$ to measure the occuring frequencies. The free energies were determined from the ensemble statistics through the Boltzmann distribution:

$$\Delta F = -k_bTlnP \qquad (2.3)$$

The clusters were classified based on finite sphere packings. All the minima in the potential-energy landscape at each value of $n < 10$ have the same potential energy, which is not the case when the potential is longer-ranged. All of the observed cluster structures agree with

theoretical predictions, ie. for $n = 2$ a dimer, a trimer for $n = 3$ and a tetrahedral cluster for $n = 4$. For higher n, multiple structures were observed. The potential energies are the same, but the entropic difference causes certain structures to be much more likely to occur than others (ie. for $n = 6$ there is the octahedron and the 'polytetrahedron', the latter occuring about 20 times more often). Guangnan et al. found that the most stable small clusters are determined by geometrical rules. First, rotational entropy favours structures with fewer symmetry elements. Second, the vibrational entropy favours nonrigid clusters (which have half-octahedral substructures sharing at least one vertex). Lastly, the potential energy favours clusters with both octahedral and tetrahedral substructures, allowing for extra bonds [11].

In 2010, Smallenburg [7] examined the formation of clusters in emulsion droplets using Monte Carlo simulations. He modelled the evaporation model in two steps. First, a compression step, where the particles are forced into a dense packing because of spherical confinement. Then a second step to minimise the second moment of mass. Only hard core interactions were taken into account. The centre of mass of each sphere modelled is confined to a spherical simulation box with hard walls. The pressure is slowly increased during the simulations from 1 to 20 in units of $\beta\sigma^3 P$. At the highest value, the particles have no more freedom to rearrange but they do vibrate. To fix this, the pressure is increased to 100, leading to a fully jammed state.

The contact angle ϑ (Young's equation) depends on three surface tensions:

$$\gamma_{\mathrm{dl}} \cos\vartheta = \gamma_{\mathrm{pl}} - \gamma_{\mathrm{pd}} \qquad (2.4)$$

where γ_{dl} is the surface tension of the droplet-liquid, γ_{pl} the surface tension of the particle-liquid and γ_{pd} the surface tension of the particle-droplet interface. In the simulations, Smallenburg considered two different regimes for particle wettability. In the non-wetting ($\cos\vartheta = 1$) regime the particles can move freely within the spherical cavity, provided they don't overlap with each other or with the droplet interface. In the finite wettability regime $-1 < cos\,\vartheta < 1$, the particles are attached to the droplet interface by an adsorption free energy. That energy is modelled by confining the centre of the mass of the particles to a thin spherical shell with a thickness of 0.1 of the spheres' size. As long as the shell thickness is a lot smaller than the size of the spheres, the exact value does not matter.

To model the second stage of evaporation, Smallenburg used jammed spherical clusters from spherical compression simulations. The spherical confinement was removed and a standard Monte Carlo scheme was used to minimise the second moment of mass M_2. In these simulations, the energy of the system is proportional to M_2, using a dimensionless proportionality constant. This constant is slowly increased to anneal the cluster to a local potential energy minimum. Both the compression and the annealing part took $8 * 10^6$ Monte Carlo cycles.

The simulations agree well with the experiments done by Manoharan et al. [5]. For four particles, a tetrahedral cluster is found, independent of the wettability [7].

Recently Peng et al. [12] used sterically stabilised dumbbell-shaped PMMA particles (these

particles have a high yield). They extended the method of Manoharan et al. [5] to generate particles with an anisotropic shape. They were interested in finding out if regular and unique structures were still formed and if the second moments of mass are still minimised. They also performed simulations to see if structures would be formed. The self assembly process was mimicked by a two step simulation procedure. Single droplets with N particles are considered through hard interactions. The particles are confined to the surface of a spherical shrinking shell in the first step. After a long time self-diffusing the particles stop and the pressure increases sharply. The second moment of mass of the particles is minimised in the second step [12].

First Peng et al. [12] applied the method to spherical particles, with cluster sizes between four and fourteen. For the values of N between four and twelve, the simulations and experiments agreed. Above twelve they did not.

The symmetric dumbbell particles have an average radius of $0.71\ \mu m$. Their polydispersity is 2.2%. The clusters were purified by density gradient centrifugation. The structures consisting of N symmetric dumbbells corresponded to structures with $2N$ spheres, for small N ($N < 5$). For higher values the structures do not coincide fully, because the dumbbells are being made of overlapping spheres, which prevents a closer packing as would be possible for spheres.

The asymmetric dumbbells consist of two spheres, one of them bigger than the other ($0.71\ \mu m$ vs. $0.52\ \mu m$). Several different structures were found for different N. As N increases, so does the amount of structures. For $N = 2$, two structures were found. For $N = 4$, eight configurations were found. The structures of the large spheres in the final configurations of the simulations agree with the experimental results, however the positions of the smaller particles vary between simulation runs and experimental snapshots [12]. The results of the simulations were predictable only if there was given less weight to the smaller spheres when minimising the second moments of mass [12].

Schwarz et al. [13] performed Monte Carlo simulations of a binary mixture of colloidal particles and spherical emulsion droplets. The colloids are modelled as hard spheres and interact via a short-ranged attraction and long-ranged repulsion. The droplet-colloid interaction is an attractive well at the droplet surface. The colloidal particles adsorb at the interface in order to minimise the interfacial free energy. This is called the Pickering effect. The droplet-droplet interaction is a hard-core interaction. To model the evaporation of the dispersed oil phase, the droplets were shrunk in time. Monte Carlo simulations are used for the dynamics of the process.

Schwarz et al. [13] also performed experiments. Polystyrene particles (154 nm in diameter) were assembled with toluene droplets as templates. The particles were analysed by cryogenic field emission scanning electron microscopy (cryo-FESEM) and field emission scanning electron microscopy (FESEM).

Before the oil evaporated, the particles are found to be disordered. The simulations confirm this finding. After the oil is evaporated, stable structures are found both in the experiments and in the simulations. Doublets, triplets, tetrahedra and more complex structures were found. For

tetrahedral clusters, there were two structures found with a different bond number, one with a bond number of 4 and one with a bond number of 6.

In the simulations, two different potentials were used. First the short-ranged attractive Asakura -Oosawa potential, and the long-ranged repulsive Yukawa potential were used. Together the authors call this the AOY potential. Secondly Schwarz et al. modelled square well interactions and the Yukawa potential, the Square-Well-Yukawa potential (SWY). It was found that although the AOY potential gives the same clusters as the SWY potential, the AOY interactions results in more possible structures, especially ones with a smaller number of bonds (more open structures). This is reasonable, because the steep attractive part of the AOY potential results in a difficult equilibration of the geometric structure of the clusters.

A mixture of tetrahedral colloids and droplets was also simulated, and superstructures were found. The octahedral dipyramid is formed by two tetrahedral clusters rotated by 30° against each other with two faces touching. The second structure contains two truncated hexagonal layers. This is only possible because one of the clusters has dissolved and is therefore not accessible experimentally when non-thermal clusters are used. The tetrahedral clusters in these experiments were thermal in the sense that the particles forming the clusters are kept together solely by short-ranged interaction and can dissolve, the bonds can break on a long time scale by thermal activation. In experiments bond breaking is unlikely because the clusters are held together by van der Waals interactions, which are stronger than the interactions used in the simulation model. The third structure that was found consists of three tetrahedral clusters. The last structure that was found was a supertetrahedron, a cluster formed by four tetrahedral clusters arranged in a tetrahedral geometry [13].

Recently, Schade et al. [14] used experiments and simulations to investigate clusters formed when colloidal spheres stick irreversibly and randomly to smaller spheres [14][15] (also called 'parking'). Oppositely charged particles are used, or particles labeled with complementary DNA sequences (the large particles do not stick to each other). The ratio α_s of large to small sphere radii is varied. Once the large spheres are bound, they cannot rearrange. This means the clusters do not form dense or symmetric packings.

In these experiments, a stoichiometric ratio of 100:1 is used, to make sure each cluster only contains one small sphere, surrounded by two or more larger spheres. After waiting several days to ensure the average cluster size is saturated, the distribution is measured.

This aggregration process yields a narrow distribution of clusters with nearly 90% of them being tetrahedra at $\alpha_s = 2.45$.

This high yield takes place in two different systems, in one where oppositely charged particles are used, and in one where the particles are labeled with complementary DNA sequences. In the first system the clustering is driven by electrostatic interactions. Large, positively-charged particles are mixed with small, negatively-charged particles. Salt is added to reduce the Debye length to approximately 3 nm, small enough to make sure the interaction range does not significantly influence the effective particle size. In the second system the clustering is driven by

the hybridisation of grafted DNA strands on the surfaces of the particles. DNA oligonucleotides are used to label the small and large spheres. The work is done well below the DNA melting temprature so the attractive interaction is many times the thermal energy k_bT.

In order to better understand why the yield is so high at $\alpha_s = 2.45$, Schade et al. performed simulations and analytical techniques. The simulations use a 'random-parking' algorithm to model the formation of clusters. The algorithm attaches large spheres to randomly generated positions on the surface of the small sphere (with no overlaps allowed). The finite range of the interactions is not modelled, because it is small compared to the particle size. The diffusion of the particles prior to binding is also not modelled. Once bound the particles can not rearrange. The process is repeated numerically to obtain distributions of cluster sizes as function of the parameter α_s.

The particles form distorted tetrahedra, since the clusters are not densely packed. The yield reaches 100% in simulations when $\alpha_s = 2.41$ is used.

The simulation results are in good agreement with the experimental results, both for the oppositely charged particles and for the particles labeled with DNA. At $\alpha_s = 2.45$, there are mainly tetramers formed. There are some trimers as well. Alternatively, at $\alpha_s = 1.90$, mainly tetramers are formed as well as a structure that includes five large spheres bound to a small sphere (both in the DNA system and in the simulations).

The fact that Schade et al. find α_c for both experimental systems and simulations, suggesting that there may be a universal origin. This arises not just because of packing constraints, but also because of the existence of a long-time lower bound that Schade et al. call the "minimum-parking" number (the minimum amount of spheres parked on a smaller sphere). They showed that there is a critical size ratio $\alpha_c = (1 + \sqrt{2}) \approx 2.41$, close to the observed point of maximum yield, where the lower bound equals the upper bound (maximum amount of spheres parked on a smaller sphere) set by packing constraints. For example, in order to have a maximum of three spheres bound, they would have to be bound such that the form a perfect circle around the smaller sphere [14]. The chance of this happening at random, is zero [7].

Experimentally, there are several methods available to make tetrahedral clusters. They can be made from several materials, such as PMMA or silica. Other clusters can also be made. In the future, these methods can be perfected to eventually make tetrahedral clusters as building blocks for a diamond lattice, on a large scale.

Chapter 3.
Theory of Monte Carlo simulations

3.1 General Theory of Molecular Simulations

In the field of molecular simulations, models are used for the particles, and for their interactions. The fundamental properties of particles, used in this work, can be summarized by their momenta and positions, denoted as $\boldsymbol{p^N}(t)$ and $\boldsymbol{r^N}(t)$ representing the momenta and positions of the N particles respectively at a time t. If we wish to determine the value of a property of a system, such as the pressure, these properties will generally depend on the positions and momenta of the N particles in the system. The value of the property A can thus be written as $A(\boldsymbol{p^N}(t), \boldsymbol{r^N}(t))$. This value $A(\boldsymbol{p^N}(t), \boldsymbol{r^N}(t))$ is defined as $A(p_{1x}, p_{1y}, p_{1z}, p_{2x}, \dots, x_1, y_1, z_1, x_2, \dots, t)$ with p_{1x} the momentum of the first particle in the x-direction and x_1 the x-coordinate of the first particle.) Over time, the property A fluctuates due to the motion of the particles. The value measured in an experiment is an average of A over time and is known as the time average. As the time increases to infinity the value of the following formula approaches the true average of the property:

$$A_{av} = \lim_{t\to\infty} \frac{1}{T} \iint_{t=0}^{T} A(\boldsymbol{p^N}(t), \boldsymbol{r}^N(t))\, dt \qquad (3.1)$$

It would then appear to be necessary to simulate the dynamic behaviour of the system. Alternatively, Boltzmann and Gibbs developed the framework of statistical mechanics, where a single system evolving in time is replaced by many copies of the system, which are no longer time-dependent. The time average then becomes an ensemble average:

$$< A >= \frac{\iint d\boldsymbol{p}^n\, d\boldsymbol{r}^n\; A(\boldsymbol{p}^N, \boldsymbol{r}^N)\rho(\boldsymbol{p}^N, \boldsymbol{r}^N)}{\iint d\boldsymbol{p}^n\, d\boldsymbol{r}^n\, \rho(\boldsymbol{p}^N, \boldsymbol{r}^N)} \qquad (3.2)$$

The double integral represents a $6N$ intergral over all positions and momenta of the particles. The probability density is denoted as $\rho(\boldsymbol{p}^N, \boldsymbol{r}^N)$, which is the probability of finding a configuration with those positions and momenta. $\iint d\boldsymbol{p}^n\, d\boldsymbol{r}^n\, \rho(\boldsymbol{p}^N, \boldsymbol{r}^N)$ is the partition function or normalisation. The ensemble average A is determined by integrating over all possible configurations of the system. The ergodic hypothesis states that the time average is equal to the ensemble average [16].

3.1.1 Derivation of the partition function

We can derive the partition function in the following way [16].

A quantum mechanical system can be found in different states. Let us limit ourselves to quantum states that are eigenvectors of the Hamiltonian $\mathcal{H}$ of the system (i.e. energy eigenstates).

For any such state $|i>$, it holds that:

$$\mathcal{H}|i> = E_i|i> \quad (3.3)$$

with E_i being the energy of state $|i>$. This is analogous to the Schrödinger equation:

$$\mathcal{H}\Psi = E\Psi \quad (3.4)$$

with Ψ the wavefunction of the system. The systems we want to simulate, contain many particles and thus the degeneracy of the energy levels is large. We define $\Omega(E, V, N)$ as the number of eigenstates with energy E of a system with N particles in a volume V. One of the basic assumptions of statistical mechanics is that a system with fixed N, V and E is equally likely to be found in any of its $\Omega(E)$ eigenstates. This is called the fundamental assumption.

Let us consider a system with total energy E, consisting of two weakly interacting subsystems. Weakly interacting here means that the subsystems can only exchange energy in the form of heat but the total energy of the system is the sum of the energies E_1 and E_2 of the subsystems. There are many ways for the energy to be distributed over the subsystems such that:

$$E_1 + E_2 = E \quad (3.5)$$

For a given choice of E_1 the total number of degenerate states of the system is:

$$\Omega(E_1, E_2) = \Omega_1(E_1) * \Omega_2(E_2) \quad (3.6)$$

The total number of states is not the sum of the number of states in the individual systems, but the product. However it is more convenient to have an additive measure of the degeneracy of the subsystems, thus we introduce the logarithm:

$$ln\,\Omega(E_1, E - E_1) = ln\,\Omega_1(E_1) + ln\,\Omega_2(E - E_1) \quad (3.7)$$

Every energy state is equally likely to occur. But the number of eigenstates corresponding to a given distribution of the energy of the subsystems depends very strongly on the value of E_1. The most likely value of E_1 is the one that maximises $ln\,\Omega(E_1, E - E_1)$. For that maximum it holds that:

$$\left(\frac{\partial \ln \Omega(E_1, E - E_1)}{\partial E_1}\right)_{N,V,E} = 0 \qquad (3.8)$$

Or:

$$\left(\frac{\partial \ln \Omega(E_1)}{\partial E_1}\right)_{N_1,V_1} = \left(\frac{\partial \ln \Omega(E_2)}{\partial E_2}\right)_{N_2,V_2} \qquad (3.9)$$

Defining:

$$\beta(E, V, N) \equiv \left(\frac{\partial \ln \Omega(E, V, N)}{\partial E}\right)_{N,V} \qquad (3.10)$$

And combining equation (2.9) and equation (2.10) leads to:

$$\beta(E_1, V_1, N_1) = \beta(E_2, V_2, N_2) \qquad (3.11)$$

If initially we put all the energy in system 1, the heat will be transferred to system 2 until equation (3.11) is satisfied. From that moment, there is (thermal) equilibrium and no further heat transfers from one system to the other. When this equilibrium is reached, $ln\,\Omega$ is at a maximum. This suggests that $ln\,\Omega$ is somehow related to the entropy S of the system, as the second law of thermodynamics states that the entropy of a system (N, V, E) is at its maximum when the system is in thermal equilibrium. For historical reasons, the entropy is defined as:

$$S(N, V, E) = k_b \ln \Omega(N, V, E) \qquad (3.12)$$

with k_b the Boltzmann constant (k_b has a value of $1.338066 * 10^{-23} J/K$). The assumption that all degenerate eigenstates of a system are equally likely implies that, in thermal equilibrium, the entropy of the system is at a maximum.

Thermal equilibrium between two systems means $\beta_1 = \beta_2$. It is also true that two bodies are in thermal equilibrium when their temperatures are equal to each other. The thermodynamic definition of temperature is:

$$\frac{1}{T} = \left(\frac{\partial S}{\partial E}\right)_{V,N} \qquad (3.13)$$

If we combine equations (2.10, 2.12 and 2.13), this leads to:

$$\beta = \frac{1}{k_b T} \qquad (3.14)$$

Let us consider what happens when we have a system A in thermal equilibrium with a large heat bath B. The total system is closed, this means that the total energy E equals:

$$E = E_B + E_A \tag{3.15}$$

and is fixed (we assume the system A and the heat bath can only exchange heat and we can ignore their interaction energy). Now suppose the system A is in a specific quantum state i with energy E_i. The heat bath then has an energy $E_B = E - E_i$ (the total energy must be constant) and the degeneracy is given by $\Omega_B(E - E_i)$. The probability P_i to find the system A in state i is determined by the degeneracy of the heat bath:

$$P_i = \frac{\Omega_B(E - E_i)}{\sum_j \Omega_B(E - E_j)} \tag{3.16}$$

To calculate $\Omega_B(E - E_i)$ we expand $ln\,\Omega_B(E - E_i)$ around $E_i = 0$:

$$ln\,\Omega_B(E - E_i) = ln\,\Omega_B(E) - E_i \frac{\partial\, ln\,\Omega_B(E)}{\partial E} + \mathcal{O}\left(\frac{1}{E}\right) \tag{3.17}$$

Combining equations (3.12, 3.13 and 3.17) leads to:

$$ln\,\Omega_B(E - E_i) = ln\,\Omega_B(E) - \frac{E_i}{k_b T} + \mathcal{O}\left(\frac{1}{E}\right) \tag{3.18}$$

If we combine (3.15 and 3.18):

$$P_i = \frac{exp\left(-\frac{E_i}{k_b T}\right)}{\sum_i exp\left(-\frac{E_i}{k_b T}\right)} \tag{3.19}$$

This is the Boltzmann distribution for a system with temperature T. We can now calculate the average energy $< E >$ of the system at a given temperature:

$$< E >= \sum_i E_i P_i$$

$$< E >= \frac{\sum_i E_i\, exp\left(-\frac{E_i}{k_b T}\right)}{\sum_i exp\left(-\frac{E_i}{k_b T}\right)} \tag{3.20)-(3.22}$$

$$< E >= -\frac{\partial \ln \sum_i\, exp\left(-\frac{E_i}{k_b T}\right)}{\partial \frac{1}{k_b T}}$$

$$< E >= -\frac{\partial \ln Q}{\partial \frac{1}{k_b T}} \tag{3.23}$$

Where we define the partition function $Q = \sum_i exp\left(-\frac{E_i}{k_b T}\right)$. If we compare equation (3.19) with the thermodynamic relation:

$$E = \frac{\partial F/T}{\partial 1/T} \tag{3.24}$$

with F the Helmholtz free energy, we see that F is related to the partition function Q:

$$F = -k_b T \ln Q = -k_b T \ln\left(\sum_i exp\left(-\frac{E_i}{k_b T}\right)\right) \tag{3.25}$$

So far we have discussed statistical mechanics in quantum mechanical terms. The entropy S is related to the density of states of a system with energy E, volume V and number of particles N. The Helmholtz free energy F is related to a partition function Q, which is a sum over all quantum states i of the Boltzmann factor $exp\left(-\frac{E_i}{k_b T}\right)$. To put this into classical terms, consider the average value of an observable A. The probability that a system at temperature T will be found in an energy eigenstate with energy E_i is known (2.19) and thus the thermal average of some observable A can be calculated:

$$< A >= \frac{\sum_i exp\left(-\frac{E_i}{k_b T}\right) < i|A|i >}{\sum_j exp\left(-\frac{E_j}{k_b T}\right)} \tag{3.26}$$

where $< i|A|i >$ denotes the expectation value of operator A in quantum state i. First the Schrödinger equation needs to be solved (3.4), then equation (3.26). Because we cannot hope to solve the Schrödinger equation for a system with many particles, we have to simplify equation (3.26) in the classical limit. It holds that:

$$exp\left(-\frac{E_i}{k_b T}\right) =< i \left| exp\left(-\frac{\mathcal{H}}{k_b T}\right) A \right| i > \tag{3.27}$$

with $\mathcal{H}$ the Hamiltonian of the system. Combining equations (3.26) and (3.27) we get:

$$< A >= \frac{\sum_i < i|exp\left(-\frac{\mathcal{H}}{k_b T}\right) A|i >}{\sum_j < j|exp\left(-\frac{\mathcal{H}}{k_b T}\right)|j >} \tag{3.28}$$

$$< A >= \frac{Tr\, exp\left(-\frac{\mathcal{H}}{k_b T}\right) A}{Tr exp\left(-\frac{\mathcal{H}}{k_b T}\right)} \qquad (3.29)$$

with Tr the trace of the operator. In linear algebra, the trace of an $n\ x\ n$ matrix is the sum of the elements on the main diagonal [17]. The value of the trace of the operator does not depend on the basis set, thus we can use any basis set we like. The Hamiltonian $\mathcal{H}$ is the sum of a kinetic part $\mathcal{K}$ and a potential energy part $\mathcal{U}$. The kinetic energy operator is a quadratic function of the momenta of all particles. This leads to the fact that momentum eigenstates are also eigenfunctions of the kinetic operator. The potential energy operator is a function of particle coordinates. Matrix elements of $\mathcal{U}$ are therefore conveniently calculated in a basis set of position eigenfunctions. However $\mathcal{H} = \mathcal{K} + \mathcal{U}$ is not diagonial in either basis set and nor is $exp\ (-\beta(\mathcal{K} + \mathcal{U})$. If we replace $exp\ (-\beta(\mathcal{H}))$ by $exp\ (-\beta\mathcal{K})exp\ (-\beta\mathcal{U})$, we could simplify equation (3.28) considerably. However this is not true because:

$$ex\, p(-\beta\mathcal{K})\, ex\, p(-\beta\mathcal{U}) = exp\left(-\beta(\mathcal{K} + \mathcal{U} + \mathcal{O}[\mathcal{K}, \mathcal{U}])\right) \qquad (3.30)$$

with $[\mathcal{K}, \mathcal{U}]$ the commutator of the kinetic and potential energy operators while $\mathcal{O}[\mathcal{K}, \mathcal{U}]$ denotes all terms containing commutators and higher-order commutators of $\mathcal{K}$ and $\mathcal{U}$. The commutator $[\mathcal{K}, \mathcal{U}]$ is of order $\hbar$ ($\hbar = h/(2\pi)$), with $\hbar$ is Planck's constant. In the limit that $\hbar \to 0$, we may ignore terms of order $\mathcal{O}[\mathcal{K}, \mathcal{U}]$. We can then write:

$$Tr\ ex\, p\big(-\beta(\mathcal{H})\big) \approx Tr\ exp(-\beta\mathcal{U})\exp\,(-\beta\mathcal{K}) \qquad (3.31)$$

If we write $|r >$ for eigenvectors of the position operator and $|k >$ for eigenvectors of the momentum operator we can write equation (3.31) as:

$$Tr\ ex\, p\big(-\beta(\mathcal{H})\big) = \sum_{r,k} < r\big|e^{-\beta\mathcal{U}}\big|r >< r|k >< k|e^{-\beta\mathcal{K}}|k >< k|r > \qquad (3.32)$$

All matrix elements can be evaluated directly:

$$< r|exp\ (-\beta\mathcal{U})|r \geq exp\ \left(-\beta\mathcal{U}(\boldsymbol{r}^N)\right) \qquad (3.33)$$

with $\mathcal{U}(\boldsymbol{r}^N)$ a function of the coordinates of all particles and not an operator.

And:

$$< k|exp\ (-\beta\mathcal{K})|k >= exp\left(-\beta \sum_{i=1}^{N} p_i^2/(2m_i)\right) \qquad (3.34)$$

with $p_i = \hbar k_i$.

$$< r|k >< k|r \geq \frac{1}{V^N} \tag{3.35}$$

with V the volume of the system and N the number of particles. Lastly, the sum over states can be replaced by an integration over all coordinates and momenta.

$$Tr\ ex\ p\big(-\beta(\mathcal{H})\big) \approx \frac{1}{h^{dN}N!}\int d\boldsymbol{p}^N d\boldsymbol{r}^N exp\left(-\beta\left(\sum_i \frac{p_i^2}{2m_i} + \mathcal{U}(\boldsymbol{r}^N)\right)\right) \equiv Q_{classical} \tag{3.36}$$

with d the dimensionality of the system. Thus the classical partition function is defined. The factor $1/N!$ has been inserted to account for the indistinguishability of the particles. Every N-particle quantum state corresponds to a volume h^{dN} in classical phase space , but not all volumes correspond to distinct states. All points in phase space that only differ in the labeling of the particles correspond to the same quantum state. Similarly the classical limit for $Tr\ exp\big(-\beta(\mathcal{H})\big)A$ can be derived and lastly the classical expression for the thermal average of A can be written as:

$$< A >= \frac{\int d\boldsymbol{p}^N d\boldsymbol{r}^N exp\left(-\beta\left(\sum_i \frac{p_i^2}{2m_i} + \mathcal{U}(\boldsymbol{r}^N)\right)\right) A(d\boldsymbol{p}^N, d\boldsymbol{r}^N)}{\int d\boldsymbol{p}^N d\boldsymbol{r}^N\ xp\left(-\beta\left(\sum_j \frac{p_j^2}{2m_j} + \mathcal{U}(\boldsymbol{r}^N)\right)\right)} \tag{3.37}$$

In the following section we will look at sampling this thermal average $< A >$.

3.1.2 Sampling

From equation (3.36) it follows that:

$$Q = c \int d\boldsymbol{p}^N d\boldsymbol{r}^N exp\,(-\frac{\mathcal{H}(\boldsymbol{r}^N, \boldsymbol{p}^N)}{k_b T}) \quad (3.38)$$

where c is a constant chosen such that the sum of quantum states in equation (3.38) approaches the classical partition function in the limit $\hbar \to 0$. For example, for a system of N idential atoms, $c = 1/(h^{3N}N!)$. The classical equation corresponding to equation (3.26) is:

$$< A >= \frac{\int d\boldsymbol{p}^n\, d\boldsymbol{r}^n\; A(\boldsymbol{p}^N, \boldsymbol{r}^N)\; exp\,(-\beta\mathcal{H}(\boldsymbol{r}^N, \boldsymbol{p}^N))}{\int d\boldsymbol{p}^n\, d\boldsymbol{r}^n\, exp\,(-\beta\mathcal{H}(\boldsymbol{r}^N, \boldsymbol{p}^N))} \quad (3.39)$$

In equation (3.39) the observable A has been expressed as a function of coordinates of the particles and their momenta. $\mathcal{K}$ is a quadratic function of the momenta and thus the integration over the momenta can be done analytically. Averages of functions depending only on momenta are therefore easy to calculate. The difficulty lies in the calculation of averages of functions $A(\boldsymbol{r}^N)$**.** In most cases this cannot be done analytically and will have to be done numerically. However because we deal with a number of particles, this would not be computationally feasable. Even if we had only 100 particles, this would take very long [16].

Let us first look at random sampling (the simplest Monte Carlo technique), before we look at Metropolis sampling [16]. We wish to evaluate a one-dimensional integral I**:**

$$I = \int_a^b dx\, f(x) \quad (3.40)$$

Equation (3.40) can be rewritten as:

$$I = (b - a) < f(x) > \quad (3.41)$$

where $< f(x) >$ is the unweighted average of $f(x)$ over the interval $[a, b]$. In brute force Monte Carlo, this average is calculated by evaluating $f(x)$ at a large number (ie. L) of values randomly distributed over the interval $[a, b]$. As $L \to \infty$, this should give the correct value for I. However, most of the computing time is spent on points where the integral is negligible. It would be preferable to sample many points where the integral is large and fewer points where it is small. This is the basic idea behind importance sampling.

Let us consider a one-dimensional example. Suppose we wish to calculate the integral (3.39) by using Monte Carlo sampling, with the sampling points nonuniformly distributed over the interval $[a, b]$ (we assume $a = 0$, $b = 1$), according to some nonnegative probability density $w(x)$. We can rewrite equation (3.40) as:

$$I = \int_0^1 dx\, w(x) \frac{f(x)}{w(x)} \tag{3.42}$$

Let us assume $w(x)$ is the derivative of another nonnegative function $u(x)$, with $u(0) = 0$ and $u(1) = 1$ (these boundary conditions mean that $w(x)$ is normalised). Then I can be written as:

$$I = \int_0^1 du\, \frac{f(x(u))}{w(x(u))} \tag{3.43}$$

In equation (3.43), $x(u)$ means that if u is the integration variable, then x must be expressed as a function of u. The next step would be to generate L random values of u uniformly distributed in the interval $[0,1]$. The estimate for I then becomes:

$$I \approx \frac{1}{L} \sum_{i=1}^{L} \frac{f(x(u_i))}{w(x(u_i))} \tag{3.44}$$

What we have gained here depends on our choice of $w(x)$. Let us estimate σ_I^2, the variance in I_L, with I_L the estimate for I obtained from equation (3.44) with L random sample points:

$$\sigma_I^2 = \frac{1}{L^2} \sum_{i=1}^{1} \sum_{j=1}^{L} \left\langle \left(\frac{f(x(u_i))}{w(x(u_i))} - \left\langle \frac{f}{w} \right\rangle \right) \left(\frac{f(x(u_j))}{w(x(u_j))} - \left\langle \frac{f}{w} \right\rangle \right) \right\rangle \tag{3.45}$$

where the angular brackets denote an ensemble average. As different samples i and j are assumed to be independent, all cross terms in equation (3.45) disappear. What is left, is:

$$\sigma_I^2 = \frac{1}{L^2} \sum_{i=1}^{L} \left\langle \left(\frac{f(x(u_i))}{w(x(u_i))} - \left\langle \frac{f}{w} \right\rangle \right)^2 \right\rangle \tag{3.46}$$

$$\sigma_I^2 = \frac{1}{L} \left(\left\langle \left(\frac{f}{w} \right)^2 \right\rangle - \left\langle \frac{f}{w} \right\rangle^2 \right) \tag{3.47}$$

Equation (3.47) shows that the variance in I still goes as $1/L$, but the magnitude of this can be greatly reduced by choosing $w(x)$ such that $f(x)/w(x)$ is a smooth function of x. Ideally, $f(x)/w(x)$ would be constant, then the variance would disappear altogether. If $w(x)$ is constant, as is the case in brute force Monte Carlo sampling, then the relative error in I can become quite large. For example, if we sample in a multidimensional configuration space of volume Ω, of which only a small fraction f is accessible, then the relative error as a result of brute force Monte Carlo sampling, is of the order $1/(Lf)$.

Because the integral in equation (3.39) is not zero only for those configurations where the Boltzmann factor is not, it would be advisable to do a nonuniform Monte Carlo sampling of configuration space, such that the weight function w is approximately proportional to the Boltzmann factor. Unfortunately, the simple scheme of importance sampling is not sufficient for this, as the step from equation (3.42) to equation (3.43) is impossible to do. In the next section we will focus on Metropolis sampling.

3.1.3 The Metropolis method

We wish to know:

$$< A >= \frac{\int d\boldsymbol{r}^n \, A(\boldsymbol{r}^N) \, exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N))}{\int d\boldsymbol{r}^n \, exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N))} \qquad (3.48)$$

What we are interested in is the ratio of these two integrals. Metropolis et al. showed that it is possible to have an efficient Monte Carlo scheme to sample this ratio [16].
To understand the Metropolis method, let us first look at equation (3.48). We will now denote the configurational part of the partition function as follows:

$$Z \equiv \int d\boldsymbol{r}^n \, exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N)) \qquad (3.49)$$

The ratio $exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N)/\, Z$ in equation (3.48) is the probability density of finding the system in a configuration around $\boldsymbol{r}^N$. Let us define the probability density:

$$\mathcal{N}(\boldsymbol{r}^n) \equiv \frac{exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N))}{Z} \qquad (3.50)$$

$\mathcal{N}(\boldsymbol{r}^n)$ is not zero. Suppose that we can randomly generate points in configuration space according to this probability density $\mathcal{N}(\boldsymbol{r}^n)$. This means that on average, the number of points n_i generated per unit volume around a point $\boldsymbol{r}^n$ is equal to $L\mathcal{N}(\boldsymbol{r}^n)$, where L is the total number of points we have generated. Or:

$$< A >\approx \frac{1}{L} \sum_{i=1}^{L} n_i \, A(\boldsymbol{r}_i^N) \qquad (3.51)$$

This equation (3.51) seems similar to equation (3.44). The difference is that in the case of (3.44) we know beforehand the probability of sampling a point in a volume $d\boldsymbol{r}^N$ around $\boldsymbol{r}^N$. In other words, we know both $exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N))$ and Z. In equation (3.51) we only know $exp\,(-\beta \mathcal{U}(\boldsymbol{r}^N))$, we only know the relative but not the absolute probability of visiting different points in configuration space.

Let us illustrate this with an example [16]. Suppose we want to measure the depth of the Nile. This could happen by conventional sampling, ie. taking points all over Egypt. This means that there will be a lot of points where the depth of the Nile is zero because there is no water. In the Metropolis method, you place yourself inside the Nile and use a random walk by using trial moves. If a move takes you out of the water, you do not accept it. After every move, the depth of the water is measured. The average of all the measurements gives you an average of the depth of the Nile.

How to generate points in configuration space with a relative probability proportional to the Boltzmann constant? The general method is to first generate the system in a configuration $\boldsymbol{r}^N$, which we denote by o (old), that has a non-vanishing Boltzmann factor $exp\ (-\beta\mathcal{U}(o))$. For example, this configuration could be a crystal with no overlaps. Next we generate a new trial configuration $\boldsymbol{r'}^N$, which we denote by n (new), by adding a small random displacement Δ to o. The Boltzmann factor of this trial configuration is $exp\ (-\beta\mathcal{U}(n))$. We must now decide if we accept or reject this new configuration.

We will now "derive" the Metropolis scheme to determine the transition probability $\pi(o \to n)$ to go from configuration o to n. It is convenient to do a thought experiment. We carry out a large number L Monte Carlo simulations parallel, with L larger than the total number of configurations. We denote the points in a configuration o by $m(o)$. We want for $m(o)$ to be on average proportional to $\mathcal{N}(o)$. The matrix elements $\pi(o \to n)$ must not destroy such an equilibrium distribution once it is reached. This means that in equilibrium, the average number of accepted trial moves that result in the system leaving configuration o must be equal to the number of accepted trial moves moving from all other configurations n to configuration o. It is useful to impose a stronger condition: in equilibrium the average number of accepted moves from o to any state n is cancelled by the number of reverse moves. This detailed balance condition means:

$$\mathcal{N}(o)\pi(o \to n) = \mathcal{N}(n)\pi(n \to o) \qquad (3.52)$$

Many possible transition matrices $\pi(o \to n)$ satisfy equation (3.52). $\pi(o \to n)$ is constructed in practice as follows. First, a trial move from state o to state n is performed. The transition matrix that determined the probability of performing a trial move from o to n is denoted as $\alpha(o \to n)$. The next state is to decide whether to accept or reject this move. The probability of accepting a trial move from o to n is denoted as $acc(o \to n)$. This means:

$$\pi(o \to n) = \alpha(o \to n)\ x\ acc(o \to n) \qquad (3.53)$$

In the original Metropolis scheme, α is chosen to be a schematic matrix ($\alpha(o \to n) = \alpha(n \to o)$). However, it is possible that α is not symmetric. If α is symmetric, equation (3.53) can be rewritten in terms of $acc(o \to n)$:

$$\mathcal{N}(o)\ x\ acc(o \to n) = \mathcal{N}(n)\ x\ \alpha(n \to o) \qquad (3.54)$$

From equation (3.54) it follows:

$$\frac{acc(o \to n)}{acc(n \to o)} = \frac{\mathcal{N}(n)}{\mathcal{N}(o)} = exp\ \Big(-\beta\big(U(n) - U(o)\big)\Big) \qquad (3.55)$$

Many choices for $acc(o \to n)$ satisfy this equation ($acc(o \to n)$ cannot exceed 1 being an obvious condition). The choice of Metropolis et al. is:

$$acc(o \to n) = \mathcal{N}(n)/(\mathcal{N}(o) \quad \text{if } \mathcal{N}(n) < N(o)$$
$$acc(o \to n) = 1 \quad \text{if } \mathcal{N}(n) \geq \mathcal{N}(o) \qquad (3.56)$$

Other choices for $acc(o \to n)$ are possible, but the original choice of Metropolis *et al.* results in a more efficient sampling than most other strategies.

In the Metropolis scheme, the transition probability going from state o to state n is given by:

$$\pi(o \to n) = \alpha(o \to n) \quad \mathcal{N}(n) \geq \mathcal{N}(o)$$
$$\pi(o \to n) = \alpha(o \to n)(\mathcal{N}(n)/(\mathcal{N}(o)) \quad \mathcal{N}(n) < N(o) \qquad (3.57)$$
$$\pi(o \to o) = 1 - \sum_{n \neq o} \pi(o \to n)$$

How to decide whether to accept or reject a trial move? Suppose we have generated a trial move from state o to state n, with $U(n) > U(o)$. According to equation (3.55) this trial move should be accepted with a probability:

$$acc(o \to n) = exp\left(-\beta\big(U(n) - U(o)\big)\right) < 1 \qquad (3.58)$$

To decide whether we accept or reject this trial move, we generate a random number $Ranf$, uniformly distributed between 0 and 1. The probabilty that $Ranf$ is less than $acc(o \to n)$ is equal to $acc(o \to n)$. We now accept the trial move if $Ranf < acc(o \to n)$ and reject it otherwise. This guarantees the probability to accept a trial move from o to n is equal to $acc(o \to n)$. It is important that the random number generator generates numbers uniformly, otherwise the simulation will be biased.

In the following section we will discuss the basic set up of a Monte Carlo simulation.

3.2 Monte Carlo simulations

The Monte Carlo method randomly generates new positions for the particles and accepts these based on a set of criteria. The probability of obtaining a certain configuration should be equal to its Boltzmann factor in the canonical ensemble:

$$exp(-\beta PU) \tag{3.59}$$

where $U(\boldsymbol{r}^N)$ is calculated using the potential energy function. In the case of hard spheres the potential is zero when the hard spheres do not touch each other and infinite when the hard spheres overlap. The average property A is calculated for each non-overlapping configuration and at the end one simply averages over the number of values calculated.

$$< A >= \frac{1}{M}\sum_{i=1}^{M} A(\boldsymbol{r}^N) \tag{3.60}$$

In a Metropolis Monte Carlo simulation [18] a new configuration is generated by randomly picking a particle, then generating a random displacement for this particle. In this case a new configuration is also found by rotating a particle (if it is a tetramer). This move is accepted based on the energy of the new configuration. However for the case of hard spheres a trial move will always be accepted unless the particles overlap.

In a Monte Carlo simulation, one has to establish first an initial configuration. The particles are placed in the simulation box. Then there is an equilibration phase, in order to equilibrate the system. The particles move and rotate. In addition, one can also perform simulations at constant pressure by allowing for volume fluctuations. If the volume changes, there is a different statistical weight for the move to be accepted. Afterwards, the production phase starts. Here the property or properties of interest are calculated, such as the volume. The system is in equilibrium but the particles still move and rotate and the volume can still change. Lastly the analysis finds place, such as the averaging over the calculated properties (ie. the volume data is now averaged over the production phase to obtain the average volume), see [18] for more details.

3.3 Periodic Boundaries and Minimum Image Convention

When a small number of particles is simulated, most of the particles are located at the surface. To counter this effect and make a bulk simulation, we use periodic boundaries [16]. This means when a particle exits the simulation box in one direction (let us say the positive x-direction) it will enter the box through the other side (in this case at the 0 in the x-direction). Minimum image convention is used, this means next to the simulation box, at all sides, there are simulation boxes with the same configuration as the main one (see Fig 3.1). When calculating the overlap between one unit and the next, the unit that is closest is taken, even if this unit is not in the main simulation box.

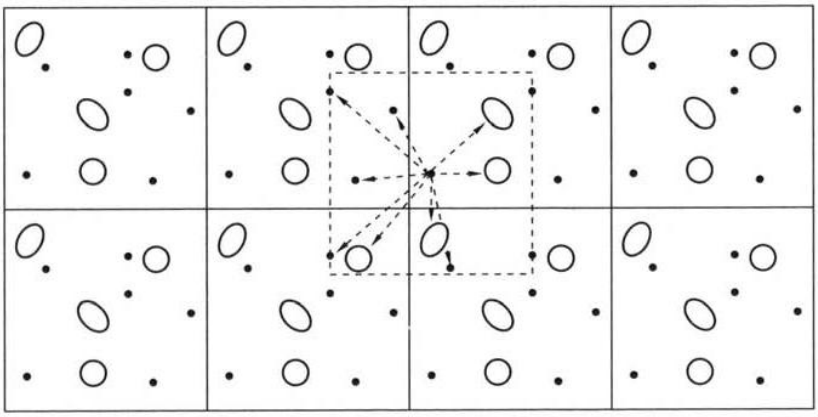


Figure 3.1: Minimum Image Convention: When calculating the overlap between two particles, the closest one is taken [16].

We use reduced units, such as $\sigma = 1$ for the diameter of the small spheres (radius $R = 0.5 * \sigma = 0.5$) and $\beta = \frac{1}{k_b T} = 1$ (with k_b is Boltzmann's constant and T the temperature).

All simulations are NPT simulations. This means that the number of particles N, the pressure P and the temperature T stay constant throughout the simulation. Certain other variables, such as the volume, vary [16].

In the next chapter we will discuss the approach used in this work, in more detail.

Chapter 4.
Simulations of tetrahedral clusters

In this chapter we will explain how the coordinate system for the four particles is set up, how the randomised rotation works and what the structure of the program is.

4.1 Definition of frames

In the simulation the program remembers the centre point of a unit. This unit may either consist of one larger sphere (see Fig. 4.1) or of a tetrahedral cluster (see Fig 4.2). If it is a tetrahedral cluster, we need to establish where the particles are positioned in relation to the centre point of the unit.

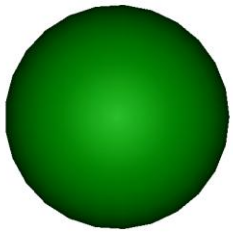


Fig 4.1: A picture of a single hard sphere.

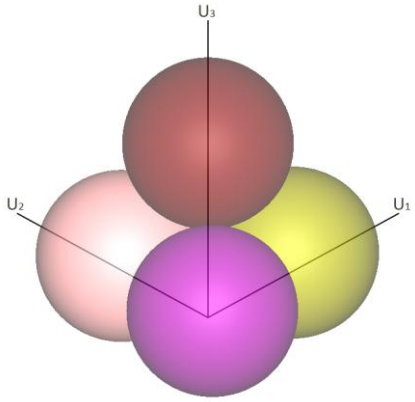


Fig 4.2: A picture of a tetrahedral cluster, four particles positioned such that they just touch each other.

For the tetrahedric structure, we will now calculate the coordinates of the centre points of its four evenly sized particles (Figure 4.3), we denote the four particles as A, B, C, and D respectively. These coordinates are also the coordinates of the vertices of a tetrahedron of which all sides are equal to 2R (two times the radius of the spheres).

First the axes are defined. The U_1-axis (x-axis) and U_2-axis (y-axis) are chosen such that one particle's centre point (the A particle) is on the U_1-axis, and the centre points of particles B and C are also in the $U_1 - U_2$ plane. When a equilateral triangle is drawn through the centre points of these three particles, the centre point of the triangle is the origin,

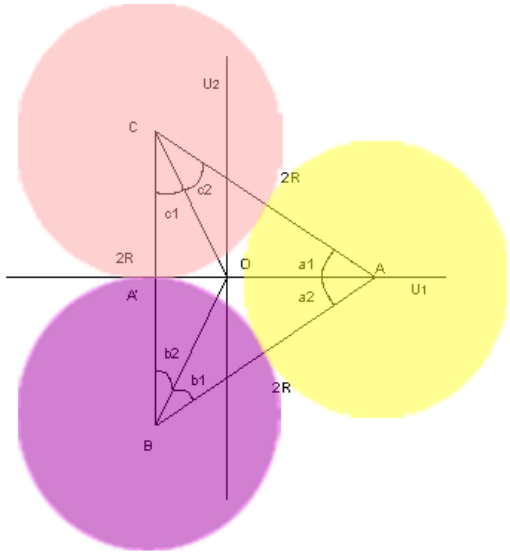


Figure 4.3: Overview of the centre points of three particles in the U_1-U_2 plane. Point A is the centre point of particle one, point B of particle two and C of particle three.

see Fig. 4.3. The last particle D is then right on top of the origin, on the U_3-axis (z-axis). The U_3-axis is positively defined in the direction of this last particle.

The particles touch each other on the sides of the given triangle and all sides are thus equal to $2R$. The angles, $a1, a2, b1, b2, c1$ and $c2$ are all 30° (half of the total angle of 60°, because it is an equilateral triangle). The point A' is in the middle of side BC. It holds that:

$$A'A = AO + A'O = \sqrt{(2R)^2 - R^2} = R\sqrt{3} \qquad (4.1)$$

It also holds:

$$A'O = \sqrt{\mathrm{CO}^2 - \mathrm{R}^2} = \sqrt{\mathrm{AO}^2 - \mathrm{R}^2} \qquad (4.2)$$

A combination of both expressions leads to:

$$AO + A'O = AO + \sqrt{\mathrm{AO}^2 - \mathrm{R}^2} = R\sqrt{3}$$

$$\sqrt{\mathrm{AO}^2 - \mathrm{R}^2} = R\sqrt{3} - AO$$

$$\mathrm{AO}^2 - \mathrm{R}^2 = AO^2 - 2R\sqrt{3}AO + 3R^2 \qquad (4.3)\text{-}(4.7)$$

$$2R\sqrt{3}AO = 4R^2$$

$$AO = \frac{4R^2}{2R\sqrt{3}} = \frac{2R}{\sqrt{3}} = \frac{2}{3}\sqrt{3}R$$

And thus follows the coordinate of the centre point of the first particle (that's on the U_1-axis): $A = (\frac{2}{3}\sqrt{3}R, 0,0)$. According to the above equations, the following holds:

$$A'O = \sqrt{\mathrm{AO}^2 - \mathrm{R}^2} = \sqrt{\frac{12}{9}\mathrm{R}^2 - \mathrm{R}^2} = R\sqrt{\frac{3}{9}} = \frac{\mathrm{R}}{\sqrt{3}} = \frac{1}{3}\sqrt{3}R \qquad (4.8)$$

From here the x-coordinates of the second and third particle follow. The y-coordinates are respectively $-R$ and R and both z-coordinates are equal to 0. From there the coordinates of these two particles follow:

$$B = (-\tfrac{1}{3}\sqrt{3}R, -R, 0); C = (-\tfrac{1}{3}\sqrt{3}R, R, 0)$$

The last particle needs a side view. In Figure 4.4 a side view is given of the particles in the $U_1 - U_3$ plane. Using Pythagoras' theorem the z-coordinate of the fourth particle can be determined.

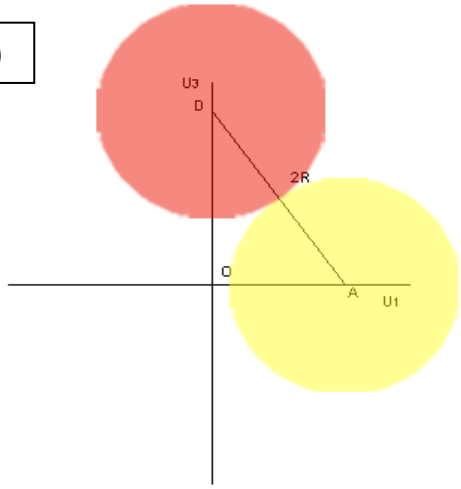


Figure 4.4: Sideview of the particles in the U_1-U_3 plane. Point A is the centre point of the first particle and point D the centre point of the last (fourth) particle.

$$\mathrm{DO} = \sqrt{(2\mathrm{R})^2 - \mathrm{AO}^2} = \sqrt{4\mathrm{R}^2 - \frac{12}{9}\mathrm{R}^2} = \sqrt{\frac{24}{9}\mathrm{R}^2} = \frac{2}{3}\sqrt{6}R \qquad (4.9)$$

The coordinates of the centre point of the last particle thus are:

$D = (0,0,\frac{2}{3}\sqrt{6}R)$.

We can summarise the calculated coordinates thus (see Table 4.1):

Particle	x	y	z
A	$\frac{2}{3}\sqrt{3}R$	0	0
B	$-\frac{1}{3}\sqrt{3}R$	$-R$	0
C	$-\frac{1}{3}\sqrt{3}R$	R	0
D	0	0	$\frac{2}{3}\sqrt{6}R$

Table 4.1 A summary of the calculated coordinates of all four particles in the coordinate frame.

4.2 Quarternions and randomised rotation

To calculate the coordinates of the four particles of a unit in a simulation, a matrix T is used. T is the matrix of the coordinates in the U_1, U_2, U_3, coordinate system U, with U always orthonormal (the three unit vectors U_1, U_2 and U_3 are perpendicular to each other). Using the coordinates calculated previously we can determine the matrix T, so each row of T contains the three (U_1, U_2, U_3) coordinates of a particle (from top to bottom A, B, C, D):

$$T = \begin{bmatrix} \frac{2}{3}\sqrt{3}R & 0 & 0 \\ -\frac{1}{3}\sqrt{3}R & R & 0 \\ -\frac{1}{3}\sqrt{3}R & -R & 0 \\ 0 & 0 & \frac{2}{3}\sqrt{6}R \end{bmatrix} \qquad (4.10)$$

We can then calculate the coordinates of the particles A-D of the tetrahedral cluster by multiplying T with the orientation matrix U (see Fig. 4.5), which converts the (x,y,z)-coordinate system into the U-coordinate system.:

$$(coordinates\ of\ particles\ (x, y\ z,)) = TU \qquad (4.11)$$

The matrix U consists of three vectors that together form the coordinate system for the particles.

$$U = [U_1 \quad U_2 \quad U_3] \qquad (4.12\text{-}4.14)$$
$$U^T U = I$$
$$||U_i|| = 1$$

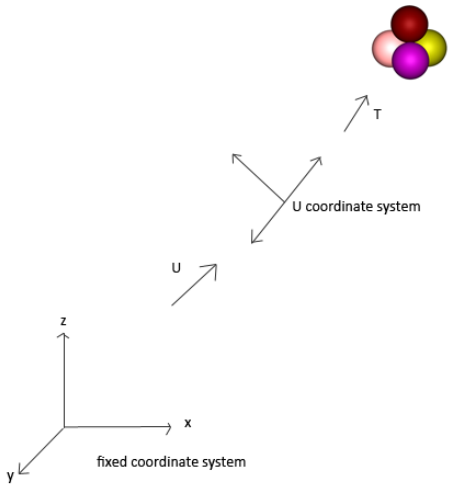


Figure 4.5: Overview of how to calculate the coordinates of the particles using the fixed coordinate system, the U coordinate system and the matrix T.

A quaternion is a four dimensional vector. We will need this vector to rotate a 3D object such as the tetrahedral cluster. A 3D object could be rotated using angles (Euler representation), but this representation could make grimbal lock happen [19]. Grimbal lock means that a rotation in one axis could 'override' a rotation in another, making you lose a degree of freedom.

A quaternion can be represented as:

$$q = q_0 + iq_1 + jq_2 + kq_3 \tag{4.15}$$

where i, j and k are imaginary units. We can use a quaternion to represent a rotation in 3D space. The multiplication identity of a quaternion is:

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{4.16}$$

which means that any quaternion multiplied by this identity quaternion, will not be changed.

To normalise a quaternion q, we use:

$$q_{norm} = \frac{q}{||q||} \tag{4.17}$$

In which $||q||$ is the magnitude of a quaternion, defined as:

$$||q|| = \sqrt{{q_0}^2 + {q_1}^2 + {q_2}^2 + {q_3}^2} \tag{4.18}$$

Quaternions used for rotational purposes are always unit quaternions. In this way, any set of points undergoes a pure rotation, rather than also a deformation. Being unit quaternions of course means that $||q|| = 1$ or:

$$q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \quad (4.19)$$

This means we can think of a rotational quaternion as a unit vector on a four dimensional sphere, where q_1, q_2 and q_3 form the axes and q_0 the angle of rotation [20]. Only unit quaternions can be used to represent rotations.

Quaternion multiplication is not communative [21][22]. This means that:

$$q_a * q_b \neq q_b * q_a \quad (4.20)$$

The inverse of a quaternion is equal to the multiplicative inverse ($1/q_a$) and can be calculated by:

$$q^{-1} = \frac{q'}{q * q'} \quad (4.21)$$

where q' is the conjugate quaternion. The complex conjugate of a number is calculated by changing all signs. This holds for any non-zero quaternion.

The quaternion way of writing down a point p is (0 degrees angle):

$$p = \begin{bmatrix} 0 \\ xi \\ yj \\ zk \end{bmatrix} \quad (4.22)$$

where x, y and z are the coordinates of point p.

The rotation around an axis is:

$$rot = \begin{bmatrix} a \\ bi \\ cj \\ dk \end{bmatrix} \quad (4.23)$$

with $a = cos\left(\frac{1}{2}\theta\right)$, $b = u_x sin\left(\frac{1}{2}\theta\right)$, $c = u_y sin\left(\frac{1}{2}\theta\right)$ and $d = u_z sin\left(\frac{1}{2}\theta\right)$, in which (u_x, u_y, u_z) represents the axes of rotation (unit vector) and θ is the angle of rotation around this axes. In order to rotate a point around an axis to get a new point p^* , we need to make the following multiplication [21]:

$$p^* = rot * p * (rot)^{-1} \quad (4.24)$$

where $(rot)^{-1}$ means the inverse of rot. We will now show the calculation for the first coordinate, xi. The calculations for the other two coordinates, yj and zk are analogous.

$$(a + bi + cj + dk)(xi)(a - bi - cj - dk) \quad (4.25)$$

In order to work this out, we need some additional information [23][24]:

$$i * j = -j * i = k$$

$$j * k = -k * j = i \quad (4.26\text{-}4.28)$$

$$k * i = -i * k = j$$

We will now work out equation (4.25):

$$(a + bi + cj + dk)(xi)(a - bi - cj - dk) = \\ a * xi * a + bi * xi * -bi + cj * xi * -cj + dk * xi * -dk + a * xi * -bi + a * xi * -cj + a * xi * -dk + bi * xi * a + bi * xi * -cj + bi * xi * -dk + cj * xi * a + cj * xi * -bi + cj * xi * -dk + dk * xi * a + dk * xi * -bi + dk * xi * -cj \quad (4.29)$$

Simplifying equation (4.29) leads to:

$$a^2xi + b^2xi - c^2xi - d^2xi + abx - acxk + adxj - abx + bcxj + bdxk - acxk + bcxj - cdx + adxj + bdxk + cdx \quad (4.30)$$

$$= a^2xi + b^2xi - c^2xi - d^2xi - 2acxk + 2adxj + 2bcxj + 2bdxk$$

The result can be divided into an x-component (xi), a y-component (xj) and a z-component (xk):

$$= (a^2 + b^2 - c^2 - d^2)xi + (2ad + 2bc)xj + (2bd - 2ac)xk \quad (4.31)$$

This holds for a point on the x-axis (where y and z are equal to zero). This multiplication is the first row of the multiplication matrix (see equation (4.32)). The multiplication can also be done for the y- and z-direction, leading to the following rotation matrix K (where $a = q_0, b = q_1, c = q_2$ and $d = q_3$) [16][19][25]:

$$K = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \quad (4.32)$$

When we want to rotate a particle, we take the orientation matrix (U) of the particle (consisting of the three vectors in the U_1-, U_2- and U_3-direction), generate a unit quaternion, generate a rotation matrix K and multiply U with K. So U is to be rotated according to:

$$U_{j+1} = K(q)U_j \tag{4.33}$$

with j being the current configuration and $j + 1$ the new, rotated configuration. K is the rotation matrix and will be determined based upon a randomly generated unit quaternion. Below follows the algorithm used to generate this four dimensional unit vector [26].

```
makevectoronunitsphere
{
	/* Declaration of variables */
	double ransq1 = 2;
	double ransq2 = 2;
	double ran1, ran2, ran3, ran4;
	double ranh;

	while(ransq1 >= 1)
	{
		ran1 = 1 - 2 * genrand_real2(); /* genrand_real2() generates a random number
between 0 and 1 */

		ran2 = 1 - 2 * genrand_real2();
		ransq1 = ran1 * ran1 + ran2 * ran2;
	}

	while(ransq2 >= 1)
	{
		ran3 = 1 - 2 * genrand_real2();
		ran4 = 1 - 2 * genrand_real2();
		ransq2 = ran3 * ran3 + ran4 * ran4;
	}

	ranh = sqrt((1 - ransq1) / ransq2); /* sqrt() calculates the square root of its given variable
*/
	temvec[1] = ran1; /* The first coordinate of the unit vector */
	temvec[2] = ran2; /* The second coordinate of the unit vector */
	temvec[3] = ran3 * ranh; /* The third coordinate of the unit vector */
	temvec[18] = ran4 * ranh; /* The fourth coordinate of the unit vector */

	return 0; /* The unit vector has been successfully generated */
}
```

We define a unit quaternion q which will be used to rotate a cluster:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \text{ and } |q| = q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1 \quad .$$

(4.34-4.37)

If $q = q_i = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$ then $K[q] = I_3$ so $U_{j+1} = U_j$. In other words, the cluster does not rotate when rotating with the identity quaternion q_i.

Since the rotation of the cluster can be very large when using a randomly generated unit quaternion, this rotation should be suppressed by using a prefactor in order to reduce it. Using λ as a prefactor $(0 \leq \lambda \leq 1)$ we can determine a new quaternion q'_{new} for rotation. In order to retain the cluster position with a prefactor of $\lambda = 0$, the identity quaternion with a prefactor of $(1 - \lambda)$ is added to the equation (4.37):

Then if $q_{new} = \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix}$ it follows that for the next q_{new}: (4.36)

$$q'_{new} = q_i (1 - \lambda) + q_{new}\lambda = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} (1 - \lambda) + \begin{bmatrix} 1/2 \\ 1/2 \\ 1/2 \\ 1/2 \end{bmatrix} \lambda = \begin{bmatrix} 1 - 1/2\,\lambda \\ 1/2\,\lambda \\ 1/2\,\lambda \\ 1/2\,\lambda \end{bmatrix} \quad (4.37)$$

q'_{new} needs to be normalised. This can be done according to equation (4.17).

4.3 Calculating the orientation using the coordinates

In the case of a tetrahedral cluster, the program saves its 'centre point' (the centre point between the particles A, B and C) and the orientation of the cluster, but the example of [3] contains coordinates instead of an orientation. In order to make the particles start the simulation in a certain structure, the orientation of these tetrahedral clusters needs to be calculated. So given coordinates of the individual particles in the fixed coordinate system (see Fig. 4.5), we try to find the orientation of the new U coordinate system.

4.3.1 Calculating the coordinates for the $MgCu_2$ structure

Here are the coordinates for the particles as given in [3], with x, y, z in the fixed coordinate system of Fig. 4.5 (see Tables (4.2-4.6). Note that in this case the particles are larger than the ones researched in this work, however the orientation matrix U is independent of this. In this section we are going to calculate the orientation vectors U_1, U_2, U_3 of the tetrahedral clusters so they are in the correct orientation in the $MgCu_2$ structure.

Set 1	x	y	z
Particle A	3.51	3.51	3.51
Particle B	3.51	1.755	1.755
Particle C	1.755	3.51	1.755
Particle D	1.755	1.755	3.51

Table 4.2: The coordinates of the first tetramer described in [3].

Set 2	x	y	z
Particle A	7.02	7.02	3.51
Particle B	7.02	5.265	1.755
Particle C	5.265	7.02	1.755
Particle D	5.265	5.265	3.51

Table 4.3: The coordinates of the second tetramer described in [3].

Set 3	x	y	z
Particle A	7.02	3.51	7.02
Particle B	7.02	1.755	5.265
Particle C	5.265	3.51	5.265
Particle D	5.265	1.755	7.02

Table 4.4: The coordinates of the third tetramer described in [3].

Set 4	x	y	z
Particle A	3.51	7.02	7.02
Particle B	3.51	5.265	5.265
Particle C	1.755	7.02	5.265
Particle D	1.755	5.265	7.02

Table 4.5: The coordinates of the fourth tetramer described in [3].

Big spheres	x	y	z
	7.8975	0.8775	0.8775
	0.8775	0.8775	0.8775
	4.3875	4.3875	0.8775
	7.8975	7.8975	0.8775
	4.3875	0.8775	4.3875
	7.8975	4.3875	4.3875
	6.1425	2.6325	2.6325
	7.8975	0.8775	7.8975
	0.8775	7.8975	0.8775
	4.3875	7.8975	4.3875
	2.6325	6.1425	2.6325
	0.8775	0.8775	7.8975

Table 4.6: The coordinates of big spheres, described in [3].

Big spheres	x	y	z
	4.3875	4.3875	7.8975
	2.6325	2.6325	6.1425
	7.8975	7.8975	7.8975
	6.1425	6.1425	6.1425
	0.8775	7.8975	7.8975
	4.3875	7.8975	4.3875

Table 4.6 (cont.): The coordinates of big spheres, described in [3].

First the centre point needs to be calculated. This point is defined as being in the middle of the particles A, B and C.

The coordinates of particle A are (x_A, y_A, z_A), the coordinates of particle B (x_B, y_B, z_B), the coordinates of particle C (x_C, y_C, z_C) and the coordinates of particle D (x_D, y_D, z_D).

The centre point E has the coordinates:

$$(x_E, y_E, z_E) = \left(\frac{x_A + x_B + x_C}{3}, \frac{y_A + y_B + y_C}{3}, \frac{z_A + z_B + z_C}{3}\right) \quad (4.38)$$

The orientation matrix U consits of three vectors, U_1, U_2, U_3 . U_1 is positioned in the direction of particle A (see Figs. 4.2, 4.3 and 4.4). U_2 is positioned next to particle B and C. U_3 is positioned in the direction of particle D.

We can calculate U_1 using:

$$U_1 = (x_A - x_E, y_A - y_E, z_A - z_E)$$

$$U_1 = \left(x_A - \frac{x_A + x_B + x_C}{3}, y_A - \frac{y_A + y_B + y_C}{3}, z_A - \frac{z_A + z_B + z_C}{3}\right) \quad (4.39)\text{-}(4.41)$$

$$U_1 = \left(\frac{2}{3}x_A - \frac{1}{3}x_B - \frac{1}{3}x_C, \frac{2}{3}y_A - \frac{1}{3}y_B - \frac{1}{3}y_C, \frac{2}{3}z_A - \frac{1}{3}z_B - \frac{1}{3}z_C\right)$$

We can calculate U_3 using:

$$U_3 = (x_D - x_E, y_D - y_E, z_D - z_E)$$

$$U_3 = \left(x_D - \frac{x_A + x_B + x_C}{3}, y_D - \frac{y_A + y_B + y_C}{3}, z_D - \frac{z_A + z_B + z_C}{3}\right) \quad (4.42)\text{-}(4.44)$$

$$U_3 = \left(x_D - \frac{1}{3}(x_A + x_B + x_C), y_D - \frac{1}{3}(y_A + y_B + y_C), z_D - \frac{1}{3}(z_A + z_B + z_C)\right)$$

We can define the separate x, y and z coordinates of the three vectors:

$$U_1 = (x_{U_1}, y_{U_1}, z_{U_1})$$
$$U_2 = (x_{U_2}, y_{U_2}, z_{U_2}) \quad (4.46)\text{-}(4.48)$$
$$U_3 = (x_{U_3}, y_{U_3}, z_{U_3})$$

All three vectors need to be perpendicular to each other, thus their inproducts equal zero.

$$x_{U_2} * x_{U_1} + y_{U_2} * y_{U_1} + z_{U_2} * z_{U_1} = 0 \quad (4.49)$$

$$x_{U_2} * x_{U_3} + y_{U_2} * y_{U_3} + z_{U_2} * z_{U_3} = 0 \quad (4.50)$$

And all vectors must have a length of 1, so:

$$\sqrt{{x_{U_2}}^2 + {y_{U_2}}^2 + {z_{U_2}}^2} = 1 \quad (4.51)$$

From these three equations (3.49-3.51) the coordinates of the vector U_2 can be calculated:

$$x_{U_2} * x_{U_1} = -y_{U_2} * y_{U_1} - z_{U_2} * z_{U_1} \quad (4.52)$$

$$x_{U_2} = \frac{-y_{U_2} * y_{U_1} - z_{U_2} * z_{U_1}}{x_{U_1}} = \frac{-y_{U_2} * y_{U_1}}{x_{U_1}} - \frac{z_{U_2} * z_{U_1}}{x_{U_1}} \quad (4.53)$$

And thus we have an expression for x_{U_2}, which is inserted into equation (4.50):

$$x_{U_3} * \left(\frac{-y_{U_2} * y_{U_1}}{x_{U_1}} - \frac{z_{U_2} * z_{U_1}}{x_{U_1}}\right) + y_{U_2} * y_{U_3} + z_{U_2} * z_{U_3} = 0 \quad (4.54)$$

We define some in between variables:

$$k = \frac{y_{U_1} * x_{U_3}}{x_{U_1}} \quad (4.55)$$

$$m = \frac{z_{U_1} * x_{U_3}}{x_{U_1}} \quad (4.56)$$

Equation (4.54) then becomes:

$$-y_{U_2} * k - z_{U_2} * m + y_{U_2} * y_{U_3} + z_{U_2} * z_{U_3} = 0$$

$$y_{U_2}(y_{U_3} - k) + z_{U_2}(z_{U_3} - m) = 0$$

$$y_{U_2}(y_{U_3} - k) = -z_{U_2}(z_{U_3} - m)$$

$$y_{U_2}(y_{U_3} - k) = z_{U_2}(m - z_{U_3})$$

$$y_{U_2} = \frac{z_{U_2}(m - z_{U_3})}{y_{U_3} - k}$$

(4.57)-(4.61)

Combining equation (4.53 and 4.61) gives:

$$x_{U_2} = -\left(\frac{(m - z_{U_3})}{y_{U_3} - k}\right) * \frac{y_{U_1}}{x_{U_1}} - z_{U_2} * \frac{z_{U_1}}{x_{U_1}} \quad (4.62)$$

Again defining an in between variable:

$$n = \frac{(m - z_{U_3})}{y_{U_3} - k} \quad (4.63)$$

Equation (4.61) then becomes:

$$y_{U_2} = z_{U_2} * n \quad (4.64)$$

Equation (4.62) then becomes:

$$x_{U_2} = -y_{U_2} * \frac{y_{U_1}}{x_{U_1}} - z_{U_2} * \frac{z_{U_1}}{x_{U_1}} \quad (4.65)$$

Combining equations (4.53, 4.61, 4.62 and 4.51) gives:

$$\sqrt{\left(-z_{U_2}\left(n * \frac{y_{U_1}}{x_{U_1}} + \frac{z_{U_1}}{x_{U_1}}\right)\right)^2 + \left(\frac{z_{U_2}(m - z_{U_3})}{y_{U_3} - k}\right)^2 + {z_{U_2}}^2} = 1 \quad (4.66)$$

Again a variable is defined:

$$p = n * \frac{y_{U_1}}{x_{U_1}} + \frac{z_{U_1}}{x_{U_1}} \quad (4.67)$$

This leads to:

$$\sqrt{\left(-z_{U_2} * p\right)^2 + \left(z_{U_2} * n\right)^2 + {z_{U_2}}^2} = 1$$

$$\left(-z_{U_2} * p\right)^2 + \left(z_{U_2} * n\right)^2 + {z_{U_2}}^2 = 1^2$$

$${z_{U_2}}^2 * p^2 + {z_{U_2}}^2 * n^2 + {z_{U_2}}^2 = 1 \quad (4.68)\text{-}(4.73)$$

$${z_{U_2}}^2(p^2 + n^2 + 1) = 1$$

$${z_{U_2}}^2 = \frac{1}{(p^2 + n^2 + 1)}$$

$$z_{U_2} = \sqrt{\frac{1}{(p^2 + n^2 + 1)}} = \sqrt{(p^2 + n^2 + 1)^{-1}} = (p^2 + n^2 + 1)^{-\frac{1}{2}}$$

Now the coordinates of the vectors U_1, U_2 and U_3 can be calculated. First E is calculated for all four sets of coordinates:

$$(x_E, y_E, z_E) = \left(\frac{x_A + x_B + x_C}{3}, \frac{y_A + y_B + y_C}{3}, \frac{z_A + z_B + z_C}{3}\right)$$

$$E_{set1} = \left(\frac{3.51 + 3.51 + 1.755}{3}, \frac{3.51 + 1.755 + 3.51}{3}, \frac{3.51 + 1.755 + 1.755}{3}\right)$$

(4.74)-(4.77)

$$E_{set1} = (2.925, 2.925, 2.34)$$

$$E_{set2} = \left(\frac{7.02 + 7.02 + 5.265}{3}, \frac{7.02 + 5.265 + 7.02}{3}, \frac{3.51 + 1.755 + 1.755}{3}\right)$$

$$E_{set2} = (6.435, 6.435, 2.34)$$

$$E_{set3} = \left(\frac{7.02 + 7.02 + 5.265}{3}, \frac{3.51 + 1.755 + 3.51}{3}, \frac{7.02 + 5.265 + 5.265}{3}\right)$$

(4.78)-(4.82)

$$E_{set3} = (6.435, 2.925, 5.85)$$

$$E_{set4} = \left(\frac{3.51 + 3.51 + 1.755}{3}, \frac{7.02 + 5.265 + 7.02}{3}, \frac{7.02 + 5.265 + 5.265}{3}\right)$$

$$E_{set4} = (2.925, 6.435, 5.85)$$

Next U_1 and U_3 are calculated.

$$U_1 = (x_A - x_E, y_A - y_E, z_A - z_E)$$

$$U_{1,set1} = (3.51 - 2.925, 3.51 - 2.925, 3.51 - 2.34) = (0.585, 0.585, 1.17)$$

$$U_{1,set2} = (7.02 - 6.435, 7.02 - 6.435, 3.51 - 2.34) = (0.585, 0.585, 1.17)$$

(4.83)-(4.87)

$$U_{1,set3} = (7.02 - 6.435, 3.51 - 2.925, 7.02 - 5.85) = (0.585, 0.585, 1.17)$$

$$U_{1,set4} = (3.51 - 2.925, 7.02 - 6.435, 7.02 - 5.85) = (0.585, 0.585, 1.17)$$

All four coordinate sets have the same U_1vector. U_3 is calculated:

$$U_3 = (x_D - x_E, y_D - y_E, z_D - z_E)$$

$$U_{3,set1} = (1.755 - 2.925, 1.755 - 2.925, 3.51 - 2.34) = (-1.17, -1.17, 1.17)$$

$$U_{3,set2} = (5.265 - 6.435, 5.265 - 6.435, 3.51 - 2.34) = (-1.17, -1.17, 1.17)$$

$$U_{3,set3} = (5.265 - 6.435, 1.755 - 2.925, 7.02 - 5.85) = (-1.17, -1.17, 1.17)$$

$$U_{3,set4} = (1.755 - 2.925, 5.265 - 6.435, 7.02 - 5.85) = (-1.17, -1.17, 1.17)$$

(4.88)-(4.92)

All four coordinate sets also have the same U_3 vector, because the three vectors are perpendicular to one another it follows that the coordinate sets must also have the same U_2vector.

Normalising U_1:

$$U_1' = \frac{(0.585, 0.585, 1.17)}{\sqrt{0.585^2 + 0.585^2 + 1.17^2}}$$

$$U_1' = (0.408, 0.408, 0.816)$$

And U_3:

$$U_3' = \frac{(-1.17, -1.17, 1.17)}{\sqrt{(-1.17)^2 + (-1.17)^2 + 1.17^2}}$$

$$U_3' = (-0.577, -0.577, 0.577)$$

Calculating the in between variables:

$$k_{set1,2,3,4} = \frac{y_{U_1} * x_{U_3}}{x_{U_1}} = \frac{0.408 * -0.577}{0.408} = -0.577$$

$$m_{set1,2,3,4} = \frac{z_{U_1} * x_{U_3}}{x_{U_1}} = \frac{0.816 * -0.577}{0.408} = -1.155$$

$$n_{set1,2,3,4} = \frac{(m - z_{U_3})}{y_{U_3} - k} = \frac{-1.155 - 0.577}{-0.577 — -0.577} = \frac{-1.732}{0}$$

(4.93)-(4.98)

This changes the formulas we started off with.

$$z_{U_2} = 0$$

$$x_{U_2} = \frac{-y_{U_1}}{x_{U_1}} y_{U_2} - \frac{z_{U_1}}{x_{U_1}} z_{U_2}$$

(4.99)-(4.101)

$$x_{U_2} = \frac{-y_{U_1}}{x_{U_1}} y_{U_2}$$

In this case y_{U_1} equals x_{U_1}, thus equation (4.101) becomes:

$$x_{U_2} = -y_{U_2} \qquad (4.102)$$

It then follows that:

$$\lambda \begin{bmatrix} -1 \\ 1 \\ 0 \end{bmatrix} \rightarrow \sqrt{(-\lambda)^2 + \lambda^2 + 0^2} = 1 \qquad (4.103)$$

$$\sqrt{2\lambda^2} = 1$$

$$\sqrt{2}\lambda = 1 \qquad (4.104)\text{-}(4.105)$$

$$\lambda = \frac{1}{\sqrt{2}} = \frac{1}{2}\sqrt{2}$$

The vector U_2 becomes:

$$U_2 = \left(-\frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{2}, 0\right) \qquad (4.106)$$

U_2 is thus equal to $\left(-\frac{1}{2}\sqrt{2}, \frac{1}{2}\sqrt{2}, 0\right)$. Because of the use of equation (4.51) this is normalised.
This can be verified:

$$\sqrt{\left(-\frac{1}{2}\sqrt{2}\right)^2 + \left(\frac{1}{2}\sqrt{2}\right)^2 + 0^2} = 1 \qquad (4.107)$$

In Table 4.7 the results have been summarised:

Set 1,2,3,4	x	y	z
U_1	0.408	0.408	0.816
U_2	$-\frac{1}{2}\sqrt{2}$	$\frac{1}{2}\sqrt{2}$	0
U_3	-0.577	-0.577	0.577

Table 4.7: The orientation vector coordinates for all four coordination sets.

The final result can be summarised as follows:

U_1 follows from equation (4.38) and equation (4.83).
U_2 follows from equation (4.66) and equation (4.78).

U_3 follows from equation (4.38) and equation (4.88).

In general there are several cases:

1. General ($x_{U_1} \neq 0$ and $y_{U_3} - k \neq 0$)
2. $x_{U_1} = 0$

$$y_{U_1} * y_{U_2} + z_{U_1} * z_{U_2} = 0 \rightarrow y_{U_2} = \frac{z_{U_1}}{y_{U_1}} z_{U_2}$$

$$x_{U_3} * x_{U_2} + y_{U_3} * \left(\frac{z_{U_1}}{y_{U_1}} z_{U_2} \right) + z_{U_3} * z_{U_2} = 0$$

$$x_{U_2} = - \frac{\left(z_{U_3} + y_{U_3} \frac{z_{U_1}}{y_{U_1}} \right)}{x_{U_3}} z_{U_2} \qquad \text{(4.108)-(4.111)}$$

$$U_2 = \lambda \begin{bmatrix} - \frac{\left(z_{U_3} + y_{U_3} \frac{z_{U_1}}{y_{U_1}} \right)}{x_{U_3}} \\ \frac{z_{U_1}}{y_{U_1}} \\ 1 \end{bmatrix}$$

3. $y_{U_3} - k \neq 0$

4.3.2 Calculating the coordinates for the fcc structure of tetrahedral clusters

To put the tetrahedral clusters in the fcc structure, the coordinates of single hard spheres in fcc are taken, and the orientation vectors U_1, U_2 and U_3 are calculated.

Here is the sets of coordinates of the first two tetrahedral clusters:

Set 1	x	y	z
Particle A	0.0	0.0	0.0
Particle B	1.0	0.0	0.0
Particle C	0.5	$\frac{1}{2}\sqrt{3}$	0.0
Particle D	0.5	$\frac{1}{3}\sqrt{3} * 0.5$	$\frac{2}{3}\sqrt{6} * 0.5$

Table 4.8: The coordinates of the first set of spheres.

Set 2	x	y	z
Particle A	2.0	0.0	0.0
Particle B	3.0	0.0	0.0
Particle C	2.5	$\frac{1}{2}\sqrt{3}$	0.0
Particle D	2.5	$\frac{1}{3}\sqrt{3} * 0.5$	$\frac{2}{3}\sqrt{6} * 0.5$

Table 4.9: The coordinates of the second set of spheres.

These particles need to be translated in the x, y and z-direction. This happens in the following way:

$$Translating: \begin{bmatrix} 2 \\ 2 * \sqrt{3} * 0.5 \\ \frac{4}{3}\sqrt{6} * 0.5 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ \sqrt{3} \\ \frac{2}{3}\sqrt{6} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (4.112)$$

Because of the way this works, the box the particles will be in, cannot be square (unless the amount of particles is infinite). We have chosen for a box of roughly 6.0 by 6.8 by 6.4, (3, 4, 4 translations in x, y and z respectively), because this gives a not too huge amount of units while also keeping the box nearly square. Within each translation there are two units, one of each set. To calculate the amount of units, we multiply the amount of translations in each direction by two:

$$units = 3 * 4 * 4 * 2 = 96 \quad (4.113)$$

Let us define the boxlength in the x-direction as b and the amount of particles in a certain direction a.

$$Boxlength\ x = b$$

$$Boxlength\ y = \frac{b}{2}\sqrt{3} \quad (4.114)\text{-}(4.117)$$

$$Boxlength\ z = \frac{b}{2} * \frac{2}{3}\sqrt{3}$$

$$\frac{b^3}{6} * \sqrt{18} = \frac{b^3}{2}\sqrt{2}$$

$$a^3 = n = \frac{b^3}{2}\sqrt{2} = \#units \quad (4.118)$$

$$units\ x = b = \neq \sqrt[3]{\frac{2}{\sqrt{2}} * \frac{b^3}{2}\sqrt{2}} = \sqrt[3]{\frac{2}{\sqrt{2}} * units} \quad (4.119)$$

We will now calculate the centre point and the orientation vectors of the two different clusters. First we calculate point E:

$$(x_E, y_E, z_E) = \left(\frac{x_A + x_B + x_C}{3}, \frac{y_A + y_B + y_C}{3}, \frac{z_A + z_B + z_C}{3}\right) \quad (4.120)$$

Filling this in for both sets:

$$E_{set1} = \left(\frac{0.0 + 1.0 + 0.5}{3}, \frac{0.0 + 0.0 + \frac{1}{2}\sqrt{3}}{3}, \frac{0.0 + 0.0 + 0.0}{3}\right)$$

$$E_{set1} = \left(0.5, \frac{1}{3}\sqrt{3} * 0.5, 0.0\right) \quad (4.121)\text{-}(4.124)$$

$$E_{set2} = \left(\frac{1.5 + 1.0 + 2.0}{3}, \frac{\frac{1}{3}\sqrt{3} * 0.5 + \frac{4}{3}\sqrt{3} * 0.5 + \frac{4}{3}\sqrt{3} * 0.5}{3}, \frac{\frac{2}{3}\sqrt{6} * 0.5 + \frac{2}{3}\sqrt{6} * 0.5 + \frac{2}{3}\sqrt{6} * 0.5}{3}\right)$$

$$E_{set2} = \left(1.5, \sqrt{3} * 0.5, \frac{2}{3}\sqrt{6} * 0.5\right)$$

Now we can calculate the orientation vectors, starting with U_1. Recall equation (4.83):

$$U_1 = (x_A - x_E, y_A - y_E, z_A - z_E) \quad (4.125)$$

Using this equation with the coordinates from Tables (4.8) and (4.9) and the values for E we just calculated:

$$U_{1,set1} = \left(0.0 - 0.5, 0.0 - \frac{1}{3}\sqrt{3} * 0.5, 0.0 - 0.0\right) \quad (4.126)$$

$$U_{1,set1} = \left(0.5, -\frac{1}{3}\sqrt{3} * 0.5, 0.0\right) \quad (4.127)$$

$$U_{1,set2} = \left(1.5 - 1.5, \frac{1}{3}\sqrt{3} * 0.5 - \sqrt{3} * 0.5, \frac{2}{3}\sqrt{6} * 0.5 - \frac{2}{3}\sqrt{6} * 0.5\right) \quad (4.128)$$

$$U_{1,set2} = \left(0.0, -\frac{2}{3}\sqrt{3} * 0.5, 0.0\right) \quad (4.129)$$

Now we need to normalise these vectors:

$$U'_{1,set1} = \frac{\left(0.5, -\frac{1}{3}\sqrt{3} * 0.5, 0.0\right)}{\sqrt{0.5^2 + \left(-\frac{1}{3}\sqrt{3}\right)^2 + 0.0^2}} = \left(-\sqrt{3} * 0.5, -0.5, 0.0\right) \quad (4.130)$$

$$U'_{1,set2} = \frac{\left(0.0, -\frac{2}{3}\sqrt{3} * 0.5, 0.0\right)}{\sqrt{0.0^2 + \left(-\frac{2}{3}\sqrt{3} * 0.5\right)^2 + 0.0^2}} = (0.0, -1.0, 0.0) \quad (4.131)$$

We can calculate U_3 using equation (4.88):

$$U_3 = (x_D - x_E, y_D - y_E, z_D - z_E)$$

$$U_{3,set1} = \left(0.5 - 0.5, \frac{1}{3}\sqrt{3} * 0.5 - \frac{1}{3}\sqrt{3} * 0.5, \frac{2}{3}\sqrt{6} * 0.5 - 0.0\right)$$

$$U_{3,set1} = \left(0.0, 0.0, \frac{2}{3}\sqrt{6} * 0.5\right) \quad (4.132)\text{-}(4.136)$$

$$U_{3,set2} = \left(1.5 - 1.5, \sqrt{3} * 0.5 - \sqrt{3} * 0.5, 0.0 - \frac{2}{3}\sqrt{6} * 0.5\right)$$

$$U_{3,set1} = \left(0.0, 0.0, \frac{2}{3}\sqrt{6} * 0.5\right)$$

We can normalise the calculated vectors:

$$U'_{3,set1} = \frac{\left(0.0, 0.0, \frac{2}{3}\sqrt{6} * 0.5\right)}{\sqrt{0.0^2 + 0.0^2 + \left(\frac{2}{3}\sqrt{6} * 0.5\right)^2}} = (0.0, 0.0, 1.0) \quad (4.137)$$

$$U'_{3,set2} = \frac{\left(0.0, 0.0, -\frac{2}{3}\sqrt{6} * 0.5\right)}{\sqrt{0.0^2 + 0.0^2 + \left(-\frac{2}{3}\sqrt{6} * 0.5\right)^2}} = (0.0, 0.0, -1.0) \quad (4.138)$$

We can now calculate U_3 using equations (4.49) and (4.50):

$$x_{U_2} * x_{U_1} + y_{U_2} * y_{U_1} + z_{U_2} * z_{U_1} = 0$$

$$x_{U_2} * x_{U_3} + y_{U_2} * y_{U_3} + z_{U_2} * z_{U_3} = 0$$

For set 1:

$$x_{U_{2,set1}} * x_{U_{1,set1}} + y_{U_{2,set1}} * y_{U_{1,set1}} + z_{U_{2,set1}} * 0.0 = 0 \quad (4.139)$$

$$x_{U_{2,set1}} * 0.0 + y_{U_{2,set1}} * 0.0 + z_{U_{2,set1}} * 1.0 = 0 \quad (4.140)$$

It then follows that:

$$z_{U_{2,set1}} = 0.0$$

$$x_{U_{2,set1}} * x_{U_{1,set1}} + y_{U_{2,set1}} * y_{U_{1,set1}} = 0.0$$

$$\sqrt{{x_{U_{2,set1}}}^2 + {y_{U_{2,set1}}}^2} = 1 \quad (4.141)\text{-}(4.146)$$

$${x_{U_{2,set1}}}^2 + {y_{U_{2,set1}}}^2 = 1$$

$${x_{U_{2,set1}}}^2 = 1 - {y_{U_{2,set1}}}^2$$

$$x_{U_{2,set1}} = \left(1 - {y_{U_2,set1}}^2\right)^{\frac{1}{2}}$$

Filling this in in equation (4.142):

$$\left(1 - {y_{U_{2,set1}}}^2\right)^{\frac{1}{2}} * x_{U_{1,set1}} + y_{U_{2,set1}} * y_{U_{1,set1}} = 0.0$$

$$\left(1 - {y_{U_{2,set1}}}^2\right)^{\frac{1}{2}} * \frac{x_{U_{1,set1}}}{y_{U_{1,set1}}} + y_{U_{2,set1}} = 0.0 \quad (4.147)\text{-}(4.150)$$

$$\left(1 - {y_{U_{2,set1}}}^2\right)^{\frac{1}{2}} = -\frac{y_{U_{2,set1}}}{\frac{x_{U_{1,set1}}}{y_{U_{1,set1}}}}$$

$$\frac{x_{U_{1,set1}}}{y_{U_{1,set1}}} = Constant = C = \sqrt{3}$$

$$1 - {y_{U_{2,set1}}}^2 = \frac{{y_{U_{2,set1}}}^2}{C^2}$$

$$1 = {y_{U_{2,set1}}}^2 \left(\frac{1}{C^2} + 1\right) \qquad \text{(4.151)-(4.154)}$$

$${y_{U_{2,set1}}}^2 = \left(\frac{1}{\frac{1}{C^2} + 1}\right) = \frac{C^2}{1 + C^2}$$

$$y_{U_{2,set1}} = \pm\left(\frac{C}{(1 + C^2)^{\frac{1}{2}}}\right) = \left(\frac{\sqrt{3}}{(1 + 3)^{\frac{1}{2}}}\right) = \sqrt{3} * 0.5$$

Filling this in in equation (4.146):

$$x_{U_{2,set1}} = \left(1 - {y_{U_{2,set1}}}^2\right)^{\frac{1}{2}} = \left(1 - \left(\sqrt{3} * 0.5\right)^2\right)^{\frac{1}{2}}$$

$$(1 - 3 * 0.25)^{\frac{1}{2}} = (1 - 0.75)^{\frac{1}{2}} = 0.25^{\frac{1}{2}} = 0.5 \qquad \text{(4.155)-(4.157)}$$

$$U_{2,set1} = \left(0.5, \sqrt{3} * 0.5, 0.0\right)$$

For set 2:

$$x_{U_{2,set2}} * 0.0 + y_{U_{2,set2}} * -1.0 + z_{U_{2,set2}} * 0.0 = 0 \qquad \text{(4.158)}$$

$$x_{U_{2,set2}} * 0.0 + y_{U_{2,set2}} * 0.0 + z_{U_{2,set2}} * -1.0 = 0.0 \qquad \text{(4.159)}$$

It follows that:

$$y_{U_{2,set2}} = 0.0$$
$$z_{U_{2,set2}} = 0.0$$

Because of normalisation (see equation (4.39)), the vector needs to have a length of 1.0, $x_{U_{2,set2}}$ must be either 1.0 or -1.0. U_2 then becomes:

$$U_{2,set2} = (\pm 1.0, 0.0, 0.0)$$

We summarise the orientation vectors for the two sets in Tables (4.10) and (4.11).

Set 1	x	y	z
U_1	$\sqrt{3} * 0.5$	0.5	0.0
U_2	0.5	$\sqrt{3} * 0.5$	0.0
U_3	0.0	0.0	1.0

Table 4.10: The orientation vectors of the first set of spheres.

Set 2	x	y	z
U_1	0.0	-1.0	0.0
U_2	± 1.0	0.0	0.0
U_3	0.0	0.0	-1.0

Table 4.11: The orientation vectors of the second set of spheres.

4.4 Structure of the program

The program consists of a main function, which addresses several sub functions. In Fig. 4.5 a flowchart of the structure of the program is shown. Below follows an explanation of all functions in the program.

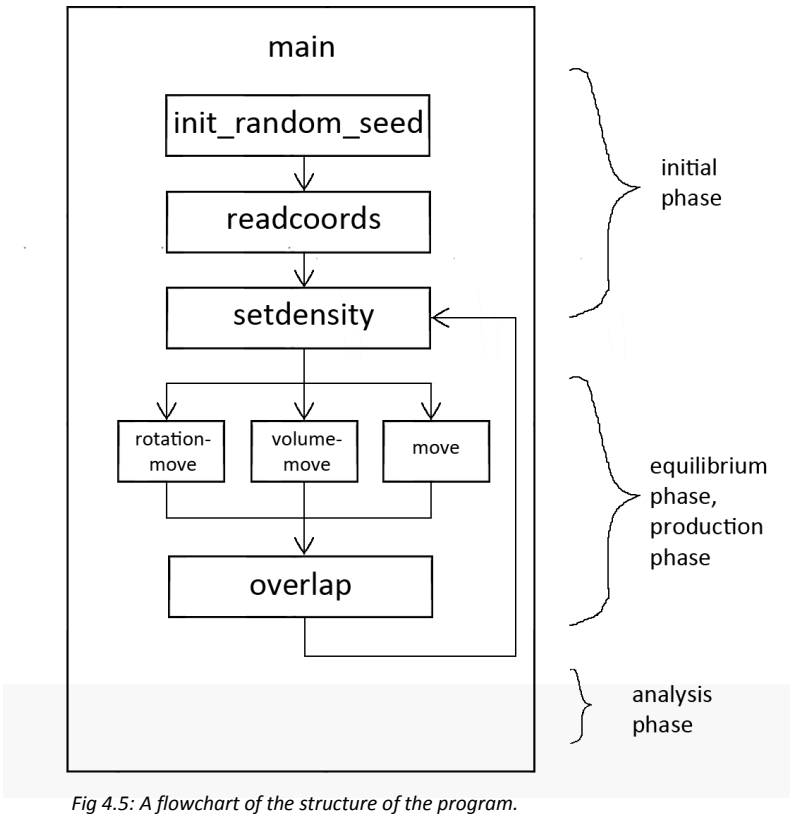


Fig 4.5: A flowchart of the structure of the program.

Main calls upon functions to initialise the number of particles (*readcoords*, *setdensity*, *writecoords_first*). It then starts the actual simulation by generating random numbers to either pick a unit to move or rotate or chooses to change the volume. The amount of random numbers generated per cycle is equal to the amount of units in the simulation. After every cycle *main* checks if the cycle number is greater than the number given for initialisation cycles and if this is the case, it writes the volume in a file and adds the volume to a variable. At the end *main* calculates the average volume from the cycles after the amount of initialisation cycles. Every fixed amount of cycles, main calls upon *writev* and *writecoords.* At the end *main* calls upon the function *writeinfo* to write information in files. It also computes the average volume from the cycles after the equilibrium phase. *Main* also calculates the percentage of accepted *moves*, *volumemoves* and *rotationmoves* as well as prints some variables.

Function *init_random_seed* seeds the random number generator. With a different seed, different random numbers are produced. If this seed would not occur, the random numbers generated would be the same for each simulation [18].

The function *overlap* checks if there is overlap between the one unit it is given and all other particles. First, it defines where the particles are, based on the centre point and (if a tetramer) the orientation. Secondly, it calculates the difference in coordinates between one particle and another, and adjusts them for minimum image convention. Finally, it tests if the cube root of the sum of its components squared is less than the radius of both particles combined (see equation 4.160).

$$\sqrt[3]{((x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2)} < (\sigma_a + \sigma_b) \quad (4.160)$$

This is done for all combinations of the given particle and any other particle, until either an overlap is found or until calculations have been made for all possible combinations of the one particle with all the other particles (ie. the function is given particle 1, it then calculates the overlap between particle 1 and all other particles in the box).

Function *readcoords* randomly places the particles in the box and calls upon *overlap*. If there is overlap the unit will be placed on a different location. R*eadcoords* also initialises the orientation matrix with set values. Alternatively, if you're doing a follow up simulation, this function reads two information files (see explanation for function *writeinfo*), takes the information from these files and uses it to build a system.

Function *setdensity* takes the value for the packing fraction and adjusts the box to be a size such that the packing fraction matches the value given. The packing fraction is defined as the volume all the particles take up, divided by the total volume and is always between zero and one. The volume of all particles is calculated in the following way:

$$V_{particles} = N_L * V_L + N_s * V_s \quad (4.161)$$

$$V_{particles} = N_L * \frac{4}{3}\pi * \sigma_L^3 + N_S * \frac{4}{3}\pi * \sigma_S^3 \quad (4.162)$$

with N_L, V_L and σ_L the number, volume and radius of the big particles and N_S, V_s and σ_s the number, volume and radius of the small particles. Note that each tetrahedral cluster contains four small particles, so: $N_S = number\ of\ tetrahedral\ clusters * 4$

Functions *writecoords_first*, *writecoords_last* and *writecoords* write the coordinates of all particles in a textbased file.

The function *writeinfo* writes some information into two files, needed for a follow up simulation. First, it writes the file for big spheres. Here the program writes down the amount of big spheres, the box lengths, the coordinates of the big spheres and their diameter (all big spheres have the same diameter, this is merely for checking). Then, it writes the file for tetrahedral clusters. Here the program prints the amount of tetrahedral clusters, the box lengths, the coordinates of the centre point of the cluster and the orientation vectors of the cluster.

Function *move* is given a unit and moves this unit. It generates a random motion with respect to its current position. It also accounts for periodic boundaries: if the new position is outside the box, the unit will enter the box from the other side, as discussed before. If there is no overlap, the unit will move to this new randomly generated position, if there is overlap the unit will remain at its old position.

Function *volumemove* changes the volume of the box with a small amount (either positive or negative). If the volume becomes negative, it will not change. The function will scale all coordinates to match the new volume. The function calls upon *overlap* and if overlap is found, the volume will not change but revert back to its old value. The new volume is accepted based on the acceptance rule:

$$rn2 > exp\left(-\beta P\sigma^3 * \left(\frac{V_{new}}{\sigma^3} - \frac{V_{old}}{\sigma^3}\right) - \beta P\sigma * u * \left(\frac{1}{\beta}\right) * log\left(\frac{V_{new}}{V_{old}}\right)\right) \quad (4.163)$$

which is dependent on the pressure P and the diameter σ. V_{new} is the new volume and V_{old} the old volume, log is the natural logarithm and u is equal to the number of units in the simulation.

Function *writev* writes the volume of the box and the amount of cycles into a text-based file. This file can be opened to see if the system was at equilibrium.

Function *makevectoronunitsphere* generates random coordinates of a 4D vector on a unit sphere.

Function *rotationmove* takes the randomly generated 4D unit vector and adjusts it with a value deltar and normalises it. The function then creates a rotation matrix K and multiplies this matrix with the orientation matrix U. This then forms a new orientation matrix (which consists of the three orientation vectors).

Chapter 5. Results of simulations

5.1 Pure hard spheres

5.1.1 Simulations of pure hard spheres

First simulations of single hard spheres were performed. In Fig. 5.1 the packing fraction versus β times the diameter of the spheres to the power of three, times the pressure is shown ($\beta\sigma^3 P$). The packing fraction is defined as:

$$\eta = \frac{N * N_{atom}}{V} \qquad (5.1)$$

where V is the average volume over the cycles after the equilibration phase in the simulation. The simulations were performed with 100 pure hard spheres, thus the packing fraction becomes:

$$\eta = \frac{100 * \frac{4}{3} * \pi * R^3}{V} \qquad (5.2)$$

where σ is the diameter of the spheres and is equal to one in the case of pure hard spheres.

For more values of simulation variables that are true in all simulations, see Table 5.1. The step sizes are a measure of how large the step should be. If this value is equal to 1, the step is the same size as dictated by the random numbers generated (we generate them between -1 and 1). This value should always be above 0, if it is 0 the new 'step' is actually the old position. By setting the value of delta at 0.3 the step will be roughly one third of the displacement (and likewise for deltav at 0.5 the step will be half the change). We aimed for choosing such step sizes, that the accepted to non-accepted move ratio is between 20-70%.

For simulation variables specific to the simulation of the pure hard spheres, see Table 5.2.

Variable	Value
Step size delta	0.3
Volume step size deltav	0.5
Beta	1

Table 5.1: Several values of simulation variables for all simulations are given.

Variable	Value
Number of particles	100
Diameter	1

Table 5.2: Several values of simulation variables for the simulations with pure hard spheres are given.

The Carnahan-Starling formula predicts the behaviour a liquid of hard spheres. The Carnahan-Starling equation is as follows [27]:

$$Z = \frac{PV}{Nk_bT} = \frac{1 + \eta + \eta^2 - \eta^3}{(1-\eta)^3} \qquad (5.3)$$

with η defined in equation (5.2).

In Figure 5.1 the equation of state for single hard spheres is plotted. The Carnahan-Starling approach is also plotted. The packing fraction η is plotted on the x-axis, $\beta\sigma^3P$ on the y-axis. Up to $\beta\sigma^3P$ the two curves are very close together. Once the hard spheres become solid in the simulation (this should happen around $\beta\sigma^3P = 11.54$ [28]), the Carnahan-Starling approach no longer gives the same results. This is as it should be.

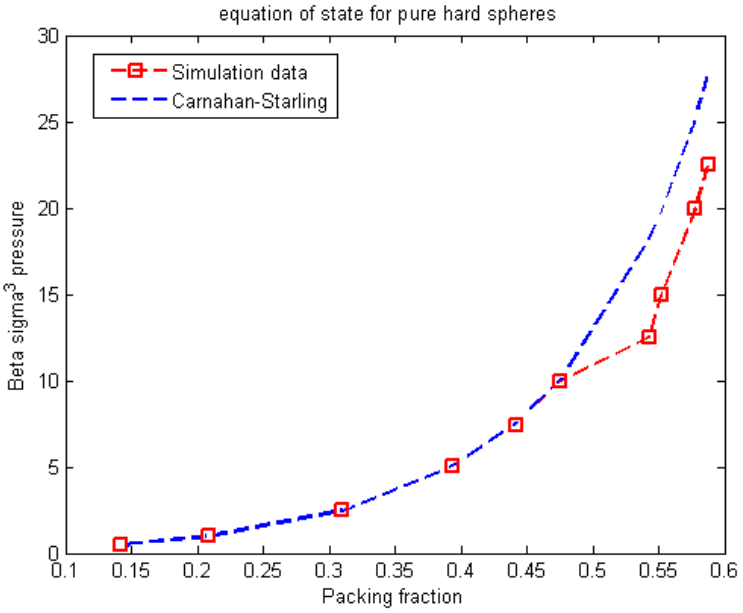


Fig. 5.1: The equation of state for pure hard spheres and the Carnahan-Starling equation of state. The x-axis is defined in units of the packing fraction, the y-axis is defined in units of βσ³P (where β equals one over temperature, σ the diameter of the spheres and P the pressure).

Figures 5.2 and 5.3 show images of low and high packing fraction respectively. Figure 5.2 was taken at the simulation with a $\beta\sigma^3P$ of 1, Figure 4.3 with a $\beta\sigma^3P$ of 15. It is clearly visible that in Figure 5.2 the particles are still liquid. In Figure 5.3 they have become a solid.

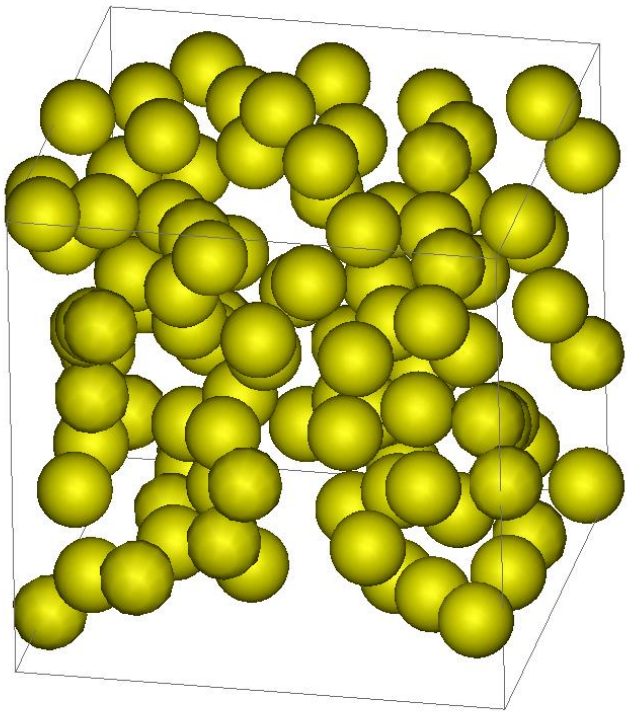


Figure 5.2: A snapshot of a simulation of pure hard spheres where $\beta\sigma^3$ P equals 1. The particles are in a disordered liquid state.

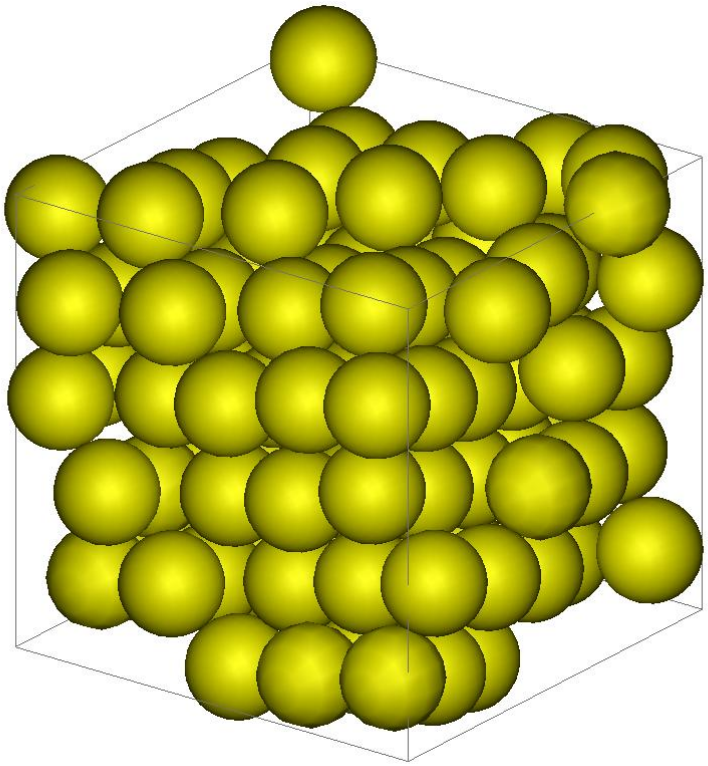


Figure 5.3: A snapshot of a simulation of single hard spheres where $\beta\sigma^3$ P equals 15. The particles are in a face-centered-cubic crystal phase.

5.1.2 Simulations based on an FCC structure

Below are the results for simulations of single hard spheres beginning in the fcc structure (face centered cubic). In Fig. 5.4 you can see a snapshot of the beginning of the simulation. In Fig. 5.5 you can see a snapshot of when the system has become a liquid. In Fig. 5.6 the equation of state is shown. You can see it is built up of two lines with a kink in the middle (where the system becomes a solid).

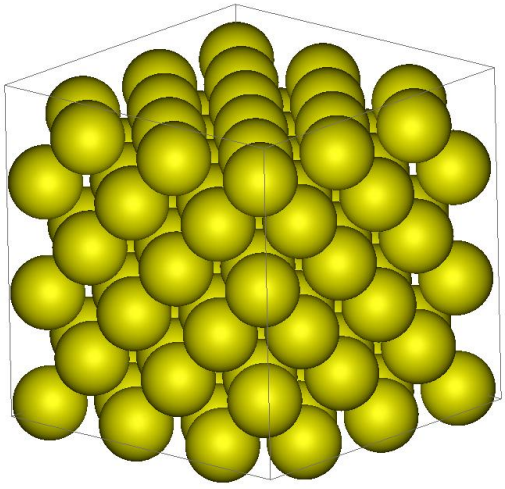


Fig. 5.4: A snapshot taken at the beginning of a simulation of single hard spheres in the fcc structure.

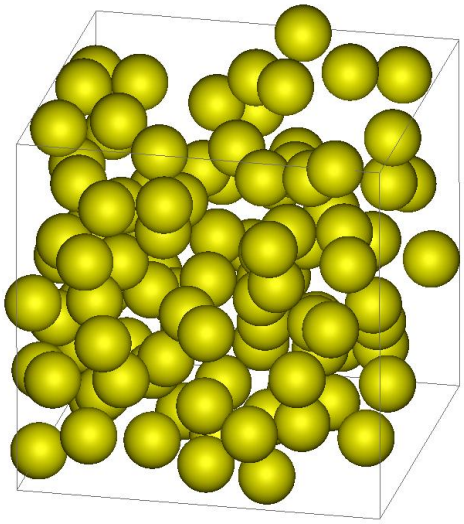


Figure 5.5: A snapshot of a simulation of single hard spheres (starting in an fcc structure) where $\beta\sigma^3 P$ equals 1. The particles are in a liquid form.

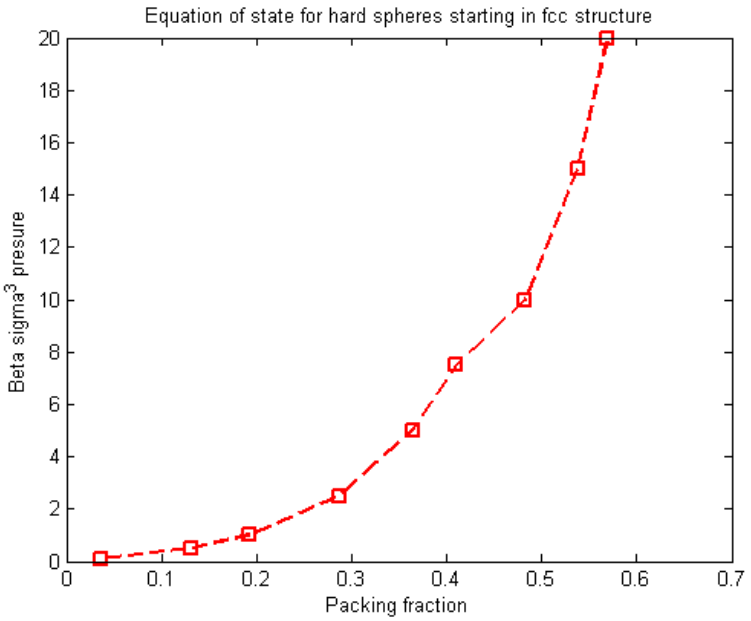


Fig. 5.6: The equation of state for single hard spheres, starting in the fcc (face centered cubic) structure. The x-axis is defined in units of the packing fraction, the y-axis is defined in units of $\beta\sigma^3 P$ (where β equals one over temperature, σ the diameter of the spheres and P the pressure).

5.2 Tetrahedral clusters

5.2.1 Simulations of a system with tetrahedral clusters

Simulations are also performed of the tetrahedral clusters (see Fig. 5.8 for an example). Fig. 5.10 shows the packing fraction versus $\beta\sigma^3 P$. First we simulated the tetrahedral clusters in an fcc structure (see Fig. 5.8). We slowly decreased the pressure and the particles turned into a liquid (see Fig. 5.9 for an example). Afterwards, we took the liquid configuration at a low $\beta\sigma^3 P$ and increased the pressure again. Each simulation contains 96 tetrahedral clusters. In total there are $96 * 4 = 384$ spheres. See Table 5.3 for values of other simulation variables. Note that when starting in the fcc structure, the box cannot be a cube or there would be space left over. This is because the amount of particles in the x-, y- and z-direction do not take up an equal amount of space. We have chosen for a system where there are three particles in the x-direction, four in the y-direction and four in the z-direction. In this system the edges of the box form a shape that is almost a cube.

Variable	Value
Amount of units	96
Amount of particles per unit	4
Rotation angle step size deltar	0.5
Diameter	1.0

Table 5.3: Several values of simulation variables for the simulations of tetrahedral clusters are given.

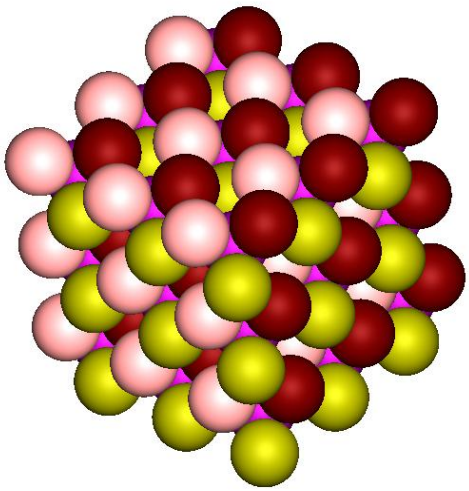


Figure 5.8: A snapshot of a simulation of tetrahedral clusters in an fcc structure.

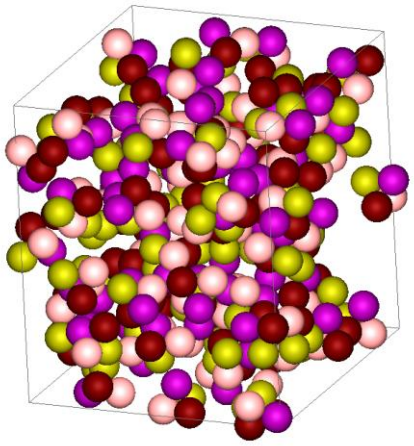


Figure 5.9: A snapshot of a simulation of tetrahedral clusters where $\beta\sigma^3 P$ equals 0.5. The particles are in a liquid form.

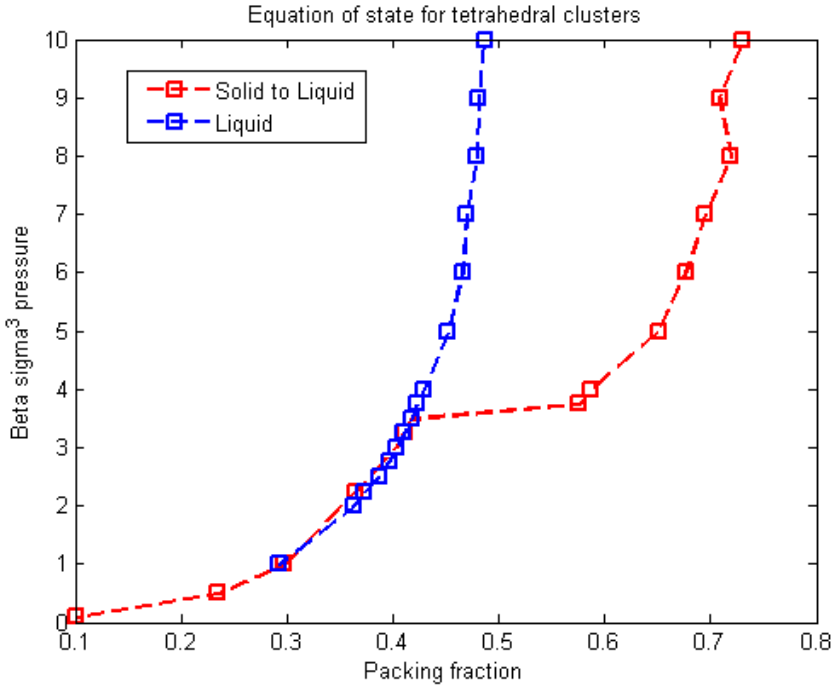


Fig. 5.10: The equation of state for tetradral clusters. The x-axis is defined in units of the packing fraction, the y-axis is defined in units of $\beta\sigma^3P$ (where β equals one over temperature, σ the diameter of the spheres and P the pressure).

In Fig. 5.10 the graph for the equation of state for tetrahedral clusters is shown. The red line denotes the values of packing fractions of the simulations where the tetrahedral clusters started in an fcc structure, then the pressure was slowly decreased. The blue line denotes the values of packing fraction of the simulations where the tetrahedral clusters began in a liquid state and the pressure was slowly increased. This graph shows somewhat similar behaviour as the hard sphere graph (see Fig. 5.6). There are in essence two lines, one for liquid and one for the solid state.

The tetrahedral clusters melt at a $\beta\sigma^3P$ of below 4. Single hard spheres melt at $\beta\sigma^3P = 11.54$ [28] . Using the law for ideal gases (equation 5.4), one can calculate that if the amount of particles is four times as little (as each cluster contains four hard sphere particles and the total amount should remain the same), the pressure will be four times as low $(11.54/4 \approx 2.89)$.

$$PV = Nk_bT \qquad (5.4)$$

5.3 Binary systems

5.3.1 Simulations of binary systems, from a random position

Simulations of the binary system were performed. In this case, the binary system consists of 20 large particles ($\sigma = \frac{1}{\sqrt{2/3}}$) and 10 tetramers ($\sigma$ of a small sphere is 1.0). In total there are $20 + 10 * 4 = 60$ particles in the system. The simulations were performed with more cycles to make sure the system had equilibriated properly. See Table 5.4 for more values of simulation variables.

Variable	Value
Amount of larger spheres	20
Amount of tetramer units	10
Total amount of particles in the box	60
Diameter of the larger spheres	$\frac{1}{\sqrt{2/3}}$
Diameter of the smaller spheres	1.0
Rotation angle step size deltar	0.5

Table 5.4: Several values of simulation variables for the simulations of the binary system are given.

Below is a graph showing the results from the simulations started from a random position (see Fig. 5.11). The σ of $\beta\sigma^3 P$ (units of the y axis) is expressed in terms of the diameter of one of the spheres of a tetrahedral cluster (so the smallest diameter and the one equal to unity). The particles are in liquid form (see Fig. 5.12), they do not seem to want to form a nice solid structure.

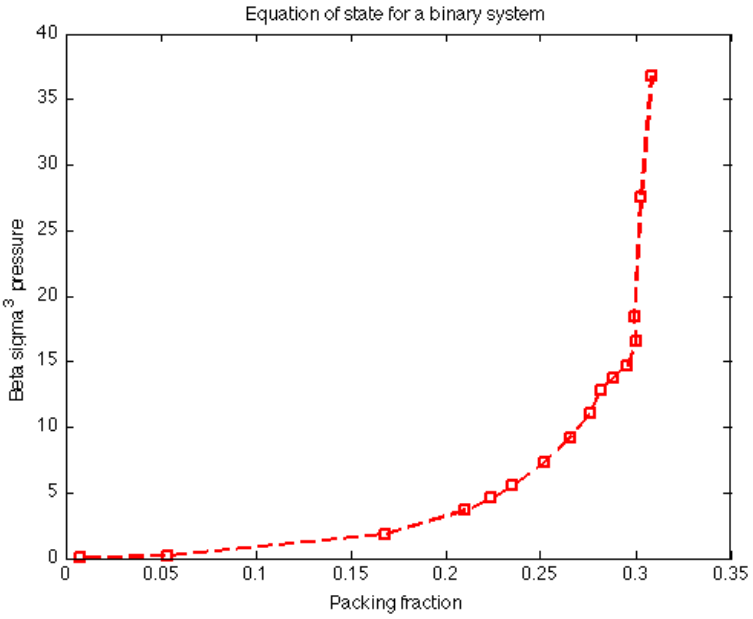


Fig. 5.11: The equation of state for binary systems. The x-axis is defined in units of the packing fraction, the y-axis is defined in units of βσ^3P (where β equals one over temperature, σ the diameter of the smaller spheres and P the pressure).

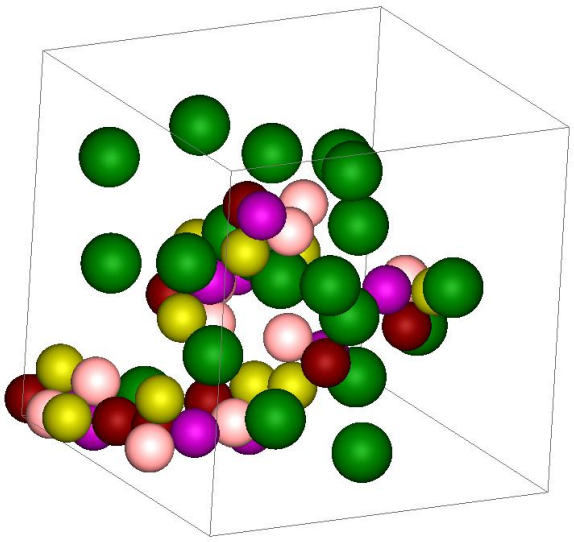


Figure 5.12: A snapshot of a simulation of a binary system where $6\sigma^3$ P equals 0.1. The particles are in a liquid form.

5.3.2. Binary systems beginning in the $MgCu_2$ structure

We took the configuration from [3] and adjusted it for our particles (this website is unfortunately no longer online). However, the packing fraction of the resulting structure could never be higher than below 0.53. The packing fraction should be 0.70, so something is wrong with the structure (see Fig. 5.13). There are too many gaps. Upon simulating, the structure immediately falls apart, indicating it's not a densest packing (as is also apparent from the value for the packing fraction).

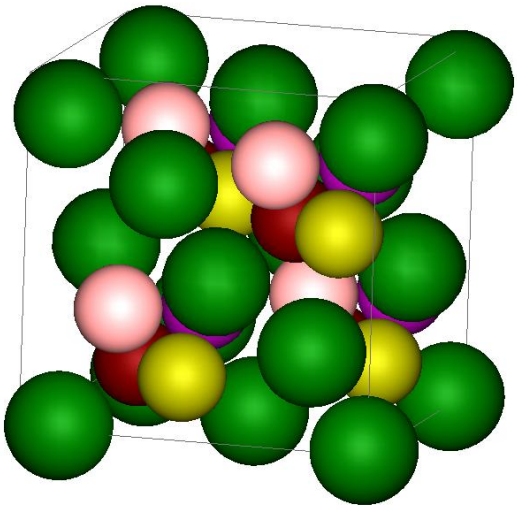


Figure 5.13: A snapshot of the beginning of a simulation, where the particles are in a $MgCu_2$ structure. From the gaps it is evident this is not a densest packing.

After months of trying different things, we eventually conclude that the original configuration [3] does have an error. If the structure is fully calculated geometrically, it should work. This is left for future research, but some ideas are given below.

A way to start with this is to draw eight big spheres on the vertices of a cube. In the middle of each side of the cube you also draw spheres (this structure is like an expanded fcc structure). Within this structure, you can see tetrahedral structures formed by the big spheres (three spheres in the middle of a cube side together with one on the vertex). In the middle of these tetrahedral structures there should be a big sphere or a tetrahedral cluster. It is the densest packing if the four particles making up the tetrahedral structure (not to be confused with a tetrahedral cluster) just touch the particle in the middle (see Fig 5.14). The distance between two spheres is then two times the radius. The edge is another variable to keep in mind. The radii of the spheres should be known, then you can calculate the edge size as a function of the radius of a sphere, using Pythagoras' theorem.

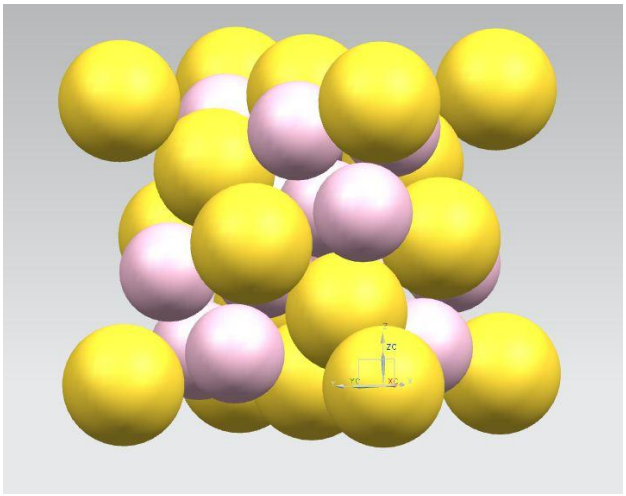


Figure 5.14: A snapshot of the $MgCu_2$ structure.

Chapter 6. Conclusion & Discussion

In this work we have programmed and performed Monte Carlo simulations of tetrahedral clusters. In the literature a variety of experimental studies have already been done, as well as a few simulation studies. When the tetrahedral clusters start in an fcc structure and the pressure is lowered sufficiently, the clusters form a liquid state. This happens at a $\beta\sigma^3 P$ of under 4, as predicted by the ideal gas law. The errorless simulations of the binary system starting in the $MgCu_2$ structure are left for future research. Monte Carlo simulations can be performed to see what happens when the pressure is lowered, where the system will melt and if there are any other phases. A phase diagram could be made. We expect it to be possible to experimentally make tetrahedral clusters on a larger scale, so that the diamond crystal can be formed and has a photonic bandgap.

References

[1] Löwen, H., Fun with hard spheres, *Statistical physics and spatial statistics: the art of analyzing and modeling spatial structures and pattern formation*: 295-331, K.R. Mecke and D. Stoyen, Lecture notes in physics No. 554, 2000.
[2] Hynninen, A. et al., Self-assembly route for photonic cyrstals with a bandgap in the visible region, *Nature*, 6: 202-205, 2007
[3] http://cst-www.nrl.navy.mil/lattice/struk/c15.html
[4] van Blaaderen, A., Colloidal Molecules and Beyond, *Science* 301: 470-471, **2003**
[5] Manoharan et al., Dense packing and Symmetry in Small Clusters of Microspheres, *Science* 301: 483-487, **2003**.
[6] Zerrouki, D., et al., Preparation of Doublet, Triangular, and Tetrahedral Colloidal Clusters by Controlled Emulsification, *Langmuir* 22: 57-62, **2006**
[7] Smallenburg, F., *Clustering and Self-Assembly in Colloidal Systems*, ISBN 9789039357088, Utrecht University, **2012**
[8] Manoharan, V., Pine, D., Building Materials by Packing Spheres, *MRS*: 91-94 **2004**
[9] Yi, G., et al., Colloidal Clusters of Silica or Polymer Microspheres, *Advanced Materials* 16, 14: 1204-1208, **2004**
[10] Cho, Y. et al., Colloidal Clusters of Microsphers from Water-in-Oil Emulsions, *Chemical Materials* 17, 5006-5013, **2005**
[11] Guangnan, M., et al., The Free-Energy Landscape of Clusters of Attractive Hard Spheres, *Science* 327: 560-563, **2010**
[12] Peng, B. et al., Colloidal Clusters by Using Emulsions and Dumbbell-Shaped Particles: Experiments and Simulations, *Angewante Chemie* 52: 1-5, **2013**
[13] Schwarz et al., Monte Carlo computer simulations and electron microscopy of colloidal cluster formation via emulsion droplet evaporation, *Journal of Chemical Physics* 135: 244501, **2011**
[14] Schade, N. et al., Tetrahedral Colloidal Clusters from Random Aggregation of Bidisperse Spheres, *Physical Review Letters* 110, 148303, 1-20, **2013**
[15] Cates, M., Self-Assembly and Entropy of Colloidal Clusters, *JCCM* 12: 1-3, **2012**
[16] Frenkel, D. & Smit, B., *Understanding molecular simulation*, Academic Press, Elsevier, **2002**
[17] http://en.wikipedia.org/wiki/Trace_%28linear_algebra%29
[18] Leach, A. R., *Molecular modelling – Principles and applications*, Pearson, Prentice Hall, **2001**
[19] http://www.gamedev.net/page/resources/_/technical/math-and-physics/quaternion-powers-r1095
[20] http://www.cprogramming.com/tutorial/3d/quaternions.html
[21] http://www.genesis3d.com/~kdtop/Quaternions-UsingToRepresentRotation.htm
[22] http://courses.cms.caltech.edu/cs171/quatut.pdf
[23] http://en.wikipedia.org/wiki/Quaternion
[24] Karney, C., Quaternions in molecular modelling, *Journal of Molecular Graphics and Modelling* 25: 595-604, **2006**
[25] Vesely, F. J., Angular Monte Carlo integration using quaternion parameters: A spherical reference potential for CCl_4, *J. of Comp. Phys.*, 47, 291-296, **1982**
[26] Marsaglia, G., Choosing a point from the surface of a sphere, *The Annals of Mathematical statistics* 43, 2, 645-646, **1972**

[27] http://www.sklogwiki.org/SklogWiki/index.php/Carnahan-Starling_equation_of_state
[28] Noya, E., et al., Determination of the melting point of hard spheres from direct coexistance simulation methods, *Journal of Chemical Physics* 128: 154507, **2008**