

AT THE UNIVERSITY OF FREIBURG

---

# Master Thesis

---

March 21, 2021

***Author:***

Wilkin Wöhler

***Supervision:***

Tanja Schilling

## Abstract

A raw writing of hard sphere nucleation, a simulation to measure quantiteis, and a analysis of data generate by means of the simulation. A test citation: [1]

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Hard sphere system . . . . .	1
1.2	metastable fluid/ phase diagram . . . . .	2
1.3	Classical nucleation theory :/ . . . . .	2
1.4	Memory including approaches . . . . .	3
1.5	Computer Precision . . . . .	4
1.6	Comparsion to Real world experiments . . . . .	4
<b>2</b>	<b>Simulation details</b>	<b>5</b>
2.1	Algorithm and Simulation details . . . . .	5
2.1.1	Event driven molecular dynamics (EDMD) . . . . .	6
2.1.2	Details of the Implementation . . . . .	8
2.2	Measurement quantities . . . . .	14
2.2.1	Mean squared displacement . . . . .	14
2.2.2	Radial distribution fucntion . . . . .	14
2.2.3	Largest Cluster . . . . .	14
2.2.4	Cluster size distribution . . . . .	14
2.2.5	Cluster positions and numbers . . . . .	15
2.2.6	Tensor of Gyration . . . . .	15
2.3	Probe of simulation code . . . . .	15
2.3.1	Diffusive behaviour . . . . .	15
2.3.2	Radial distribution function . . . . .	17
2.4	Estimate of required resources . . . . .	18
2.4.1	Calculation time estimates . . . . .	19
2.4.2	File sizes estimates . . . . .	20
2.5	First produced data . . . . .	21
2.6	Possible extensions . . . . .	24
2.6.1	Varying radius . . . . .	25
2.6.2	Varying mass . . . . .	25
2.6.3	Multiprocessing . . . . .	25

<b>3</b>	<b>Data Analysis</b>	<b>26</b>
3.1	Cluster growth . . . . .	26
3.2	Tensor of Gyration properties . . . . .	26
3.3	ACF largest cluster? . . . . .	26
3.4	Nucleation time dilemma . . . . .	26
3.4.1	induction time by critical cluster from cnt . . . . .	26
3.4.2	inductio time by constant growth extrapolated to zero . . . . .	26
3.4.3	induction time by exponential fit . . . . .	27
3.4.4	indutcion time by fluctation horizon crossing . . . . .	27
3.5	Induction time by exponential distribution . . . . .	27
3.6	Nucleation rate comparison . . . . .	27
3.7	Memory Kernels . . . . .	27
<b>4</b>	<b>Conclusion - Summary</b>	<b>28</b>
4.1	Conclusion . . . . .	28
<b>5</b>	<b>Appendix</b>	<b>29</b>
.1	A . . . . .	29

# List of Figures

2.3.1	Logarithmic long time diffusion constant of the hard sphere liquid as measured in the own simulations as well as measurements from the literature. <a href="#">cite the people</a> . . . . .	16
2.3.2	Radial distribution functions for a range of volume fractions. The colouring corresponds to the used volume fraction. . . . .	17
2.3.3	Radial distribution function for the hard sphere system at a low and at a high volume fraction of the liquid together with the theoretical prediction from the Percus-Yevick approximation. . . . .	18
2.4.1	Overview of CPU time required for calculating a simulation step, consisting of an event for each particle, and a measurement of relevant quantities for the system. As assumed for a simulation algorithm with $\mathcal{O}(N)$ calculation effort, the data points can be described by a line rather well. As the CPU time is clearly related to the further workload of the CPU during the calculation it is also expected to find fluctuations if the other workload of the machine is not strictly controlled. . . . .	19
2.4.2	Calculation time of the q6q6 order parameter at an increasing largest cluster size during one nucleation, together with the quadratic best fit indicating that the q6q6 routine calculation effort can be approximated by $\mathcal{O}(N_{lc}^2)$ where $N_{lc}$ is the size of the largest cluster. . . . .	20
2.4.3	Overview of filesizes when a single setup on the one hand and a single full simulation on the other hand is saved for comparison reasons together with their corresponding linear regression. while the linear regression for 3 points is statistically not exceedingly senseful it still remains a useful tool to extract the slope which corresponds to the required memory per particle and snapshot or reset simulation. . . . .	21
2.5.1	Heat map of the mean cluster distribution over time. The diagram encompasses 800 trajectories of 16384 particles each. The colouring indicates the logarithm of the mean cluster occurance corresponding to a probability in the stationary case. . . . .	22
2.5.2	Heat map of differences between the cluster distributions within simulations carried out with varying the length of the equilibration phase. The quantity used for colouring is defined in Equation 2.5.1, where yellow indicates a large difference while blue indicates a small difference. Providing a legend of the colouring is omitted as $\Delta_{p(N,t)}$ has no further use as to indicate differences and actual values do not add any use. . . . .	23

2.5.3 Heat map of differences between the cluster distributions within simulations carried out with varying the volume fraction of the liquid during the equilibration phase. The quantity used for colouring is defined in Equation 2.5.1, where yellow indicates a large difference while blue indicates a small difference. Providing a legend of the colouring is omitted as $\Delta_{p(N,t)}$ has no further use as to indicate differences and actual values do not add any use. . . . .	24
2.5.4 Comparison of nucleation rates under CNT assumptions for different initial densities during equilibration with eq_steps fixed at 5000, as well as varying eq_steps with $\eta_i$ fixed at 0.45. . . . .	25

# List of Tables

2.1.1 Content of the <i>Event</i> struct. . . . .	8
2.1.2 Possible results for left and right crossing time with resulting choice of next crossing time. $>$ , $=$ and $<$ are to be read as for example $t_1 > 0$ . The case indicates the case number within the actual simulation. . . . .	11
2.1.3 Overview of the cells' <i>neighbours</i> indices directly sharing a surface for 3 dimensions. As the indices hardly follow any simple pattern they are explicitly noted at this point. Obviously the cell consists of a front and a back boundary in each dimension. The corresponding case matches the one from Table 2.1.2. . . . .	12
2.3.1 Input parameters of diffusion test systems. . . . .	17
2.5.1 Input parameters of test systems probing the dependence on equilibration steps and initial density. . . . .	22



# 1 Introduction

## 1.1 Hard sphere system

Some test text.

The Hard Sphere system is the simplest model of a fluid including interactions between the single particles. Its well known potential between particles  $i$  and  $j$  reads:

$$V(r_{ij}) = \infty \cdot \Theta(\sigma - r_{ij}) \quad (1.1.1)$$

In this equation  $r_{ij}$  indicates the distance between the two particles,  $\sigma$  is the diameter of the Hard Spheres and  $\Theta$  is the Heavyside function.

While the ideal gas model without pair interactions already makes it possible to derive famous equations as  $pV = NkT$ , it does not include phase transitions yet. But these can be observed when granting the particles to take up space. Because it is the simplest model and it is well feasible for computer simulations the Hard Sphere system is very well suited to study basic properties of phase transitions.

Compared to experiments where similar systems can also be realized, general properties of the system at hand can be varied very precisely without much effort and position data of the single particles can be extracted easily as well, because they naturally are required for the simulation.

On the downside computer simulations are much more constraint in their size, but with todays computational possibilities system of the order of 1 million particles become tractable, and such computer simulations become a powerfull tool to study also phase transitions in simple systems.

The beginning of such simulations actually dates back to the beginning of electronic computer technology (cite Alder and Wainwright 1959). Since then more algorithms to increase efficiency have been elaborated, and technology advanced giving today the possibility of studying large systems.



## 1.2 metastable fluid/ phase diagram

The equation of state for the simple Hard Sphere system has various approximations, (cite an overview paper), The probably most common approximation due to its simplicity is the Caranhan-Sterling approximation:

$$Z = \frac{1 + \eta + \eta^2 - \eta^3}{(1 - \eta)^3} \quad (1.2.1)$$

(cite <https://aip.scitation.org/doi/10.1063/1.1672048>) It approximates the compressibility factor  $Z$  depending on the packing fraction  $\eta$  for the Hard Sphere fluid.

For the stable brach after nucleation a common approximation is given by the Alamrza equation of state( cite <https://aip.scitation.org/doi/full/10.1063/1.3133328> ).

$$\frac{p(v - v_0)}{k_B T} = 3 - 1.807846y + 11.56350y^2 + 141.6y^3 - 2609.26y^4 + 19328.09y^5 \quad (1.2.2)$$

where  $p$  is the pressure,  $v$  is the volume per particle  $v_0 = \sigma^3/\sqrt{2}$  is the volume per particle at close packing, including the diameter of the spheres  $\sigma$ , and  $y = p\sigma^3/(k_B T)$ , where  $k_B$  is the Boltzman constant and  $T$  is the temperature of the crystal.

From these two equations of state we can draw the phase diagram: (include grapic of phase diagram.)

The chemical potential difference between the two equations of state can be calculate from the difference between the two equations of state.

Eventhough not further discussed in this thesis it might be said that for polydisperse radii the phase diagram becomes even richer as show for example in <https://journals.aps.org/prl/pdf/10.1103/PhysRevLett.91.06830>

## 1.3 Classical nucleation theory :/

Classical nucleation theory (CNT) has been proposed [look who did this the first time](#) and since then multiple times modified to describe various types of systems. Eventhough its predictions often deviate far from experimental results. [look if tanja cited someone else here](#). Either way it can provide a reference to know what to expect roughly and also it can be checked by the simulation data.

CNT assumes that a spherical crystalite may form in the liquid with properties of the bulk crystal while the fluid remains with the properties of the bulk liquid. The difference in the free energy landscape is given by a surface and a volume term. The first arises from the surface tension  $\gamma$  between the fluid bulk and solid bulk phases. The second comes by the difference in chemical potential  $\Delta\mu$ . The whole

expression reads:

$$\beta\Delta G(R) = 4\pi R\gamma - \frac{4}{3}\pi R^3\rho\Delta\mu \quad (1.3.1)$$

Where  $\rho$  is the particle density of the solid phase.

The free energy barrier can be sketched like this (maybe sketch it for some  $\mu$  and  $\gamma$ ).

As can be seen the free energy landscape of eq. 1.3.1 has a maximum at a radius called  $R_{crit}$ . The interpretation of this radius is that if a cluster, which is defined as a structure having a crystalline like ordering locally, surpasses the critical radius, it is likely to continue to grow until it incorporates all available fluid. This critical radius is given by eq. 1.3.2.

$$R_{crit} = \frac{2\gamma}{\rho\Delta\mu} \quad (1.3.2)$$

Furthermore the height of this barrier can be calculated to be

$$\beta\Delta G(R_{crit}) = \frac{16\pi\gamma^3}{3\rho^2(\Delta\mu)^2} \quad (1.3.3)$$

In the classical picture now we look at an activated process which has a rate given by the Arrhenius law:

$$k = c \exp\left(-\frac{k_B T}{\Delta E}\right) \quad (1.3.4)$$

$$\Leftrightarrow k = c \exp -\beta\Delta G(R_{crit}) \quad (1.3.5)$$

As the constant  $c$  is not further defined the absolute nucleation rate in this picture is not predicted but only set into relation to other rates. **Check carefully in how far this is correct!**

Given the equation of state for the liquid and fluid phase from section 1.2, and taking a literature value for  $\gamma$  **cite the reference to which the value corresponds** we can calculate  $R_{crit}$  for various densities.

**Plot of  $r_{crit}$  for different densities**

## 1.4 Memory including approaches

CNT assumes a Markovian system, which means that it diffuses through a free energy landscape without granting it memory. But it has been shown by Kuhnbold ... **cite Anjas Lennard-Jones System** that for a Lennard-Jones system memory effects can not be neglected if an accurate description is desired. Such it must be assumed that also for the Hard Sphere system memory effects may be present.

To analyze such memory effects a framework by Hugues Meyer (**cite Hugues**) has been elaborated. In

it a memory kernel for coarse-grained observables by means of projection operators is defined together with an efficient algorithm to calculate such.

With the derived memory kernel an equation of motion for the observable can be dervied, ressmbling mostly the Generalized Langevin equationbu is called non-stationary Generalized Langevin Equation because the memory kernel can depend not only on  $K(t - \tau)$  but instead on two times  $K(t, \tau)$ :

$$\frac{dA_t}{dt} = \omega(t)A_t + \int_0^t d\tau K(\tau, t)A_\tau + \eta_t \quad , \quad (1.4.1)$$

For the Markovian process the memory kernel is approximately given by a Dirac delta functional  $\delta(\tau - t)$ , in which case the Langevin equation is recovered.

## 1.5 Computer Precision

The precision of computer certainly influences the outcome of simulations. (talk with fabian if he found any papers reagarding varying results fromvarying precision over time)

And it should always be kept in mind that the simulation only approximates the real world. Even smallest variations in the last digits of positions, changes the simulation radical after a certain number of steps. This comes by the fact that we face a complex system with chaotic behaviour, which means that even small variations will grow exponentially until the system has nothing in common anymore. Look if you can visulaize such an behaviour by As it can be seen after x timesteps the two system have radically changed.

## 1.6 Comparsion to Real world experiments

Since the 1950s ? people have syththesized hard sphere like systems in the lab. Today a whole zoo of systems is known. All of these system have in common, that the hard spheres are in a bath of a fluid, which surrounds them. This fluids density and optical refractive index shhould match the density and optical refractive index of the hard spheres to prevent segmentation and to enable optical measurements of the system. maybe elaborate a little more on why each of the two is necessary

The absence of the bath in simple hard sphere simulations is probably the largest difference to the hard sphere systems in the laboratory. It has been argued that the difference can be circumvented by normalizing with a diffusion length or time, but a discussion on the possibility of hydrondynamic effects changing the behaviour of the lab system compared to simulations is ongoing at the moment.

For simulations it is much harder to include the bath becuase it introudces a multiple number of particles, making it very slow to simulate large systems.

## 2 Simulation details

During the course of the master thesis an event driven molecular dynamic (EDMD) simulation code has been elaborated. The choice to use the EDMD approach is taken because interest in the actual dynamics of the system were desired. This means that simulations probing the phase space of the system instead of the dynamics, like Monte Carlo (MC) simulation schemes, are not suited.

Furthermore the discontinuous potential of the hard spheres is an obstacle not easy to face in regular molecular dynamics (MD) schemes, where the Newtonian equation of motion for the particles is numerically integrated.

The EDMD approach on the other side actually requires these discontinuities as will be discussed in the following sections, together with some details of the program.

### 2.1 Algorithm and Simulation details

In this section we will highlight the main differences to regular MD simulations, as they are the main tool to otherwise probe the dynamics of the system. Furthermore we will stick to the hard sphere example when discussing the EDMD simulations, but it can be kept in mind that the EDMD approach also allows to simulate particles with other potentials as long as the potentials are only containing step functions.

The decisive difference between EDMD simulations and regular MD schemes is that, instead of evaluating all pair and external forces on each particle and then evolving the whole system to the next time step, EDMD simulations do not have a predefined time step, but the system is evolved from one event to the next one. An event in this context is defined as the time where the next collision in the whole system takes place.

The event prediction algorithm follows closely the approach proposed by Bannerman et. al [2] which will be discussed in the next section.

### 2.1.1 Event driven molecular dynamics (EDMD)

For the prediction of events in EDMD simulations an overlap function  $f_{ij}(t)$  between particles  $i$  and  $j$  is defined, where the squared quantities are used merely because they are easily accessible.

$$f_{ij}(t) := |\vec{r}_j(t) - \vec{r}_i(t)|^2 - \sigma^2 \quad (2.1.1)$$

$$\left| \begin{array}{l} \text{with } \vec{r}_i(t) = \vec{r}_i(t_0) + (t - t_0) \vec{v}_i(t_0), \\ \Delta t := t - t_0, \\ \vec{v}_{ij}(t) := \vec{v}_j(t) - \vec{v}_i(t), \\ \vec{r}_{ij}(t) := \vec{r}_j(t) - \vec{r}_i(t), \\ \Leftrightarrow \vec{r}_{ij}(t) = \vec{r}_{ij}(t_0) + \Delta t \vec{v}_{ij}(t_0) \end{array} \right. \quad (2.1.2)$$

$$f(t) = (\vec{r}_{ij}(t_0) + \Delta t \vec{v}_{ij}(t_0))^2 - \sigma^2 \quad (2.1.3)$$

$$f(t) = |\vec{r}_{ij}(t_0)|^2 + \Delta t^2 |\vec{v}_{ij}(t_0)|^2 - 2\Delta t \vec{r}_{ij}(t_0) \cdot \vec{v}_{ij}(t_0) - \sigma^2 \quad (2.1.4)$$

The overlap function has the property that it is negative for two particles being closer than their diameter, 0 for at collision and positive if neither overlapping nor touching. The calculation of the next collision thus is to calculate the roots of eq. 2.1.4.

Solving for  $\Delta t$  with  $|\vec{r}_{ij}(t_0)|^2 := rr$ ,  $|\vec{v}_{ij}(t_0)|^2 := vv$  and  $\vec{r}_{ij}(t_0) \cdot \vec{v}_{ij}(t_0) := rv$  is rather trivial:

$$0 = rr + vv \Delta t^2 - 2rv \Delta t - \sigma^2 \quad (2.1.5)$$

$$\Leftrightarrow 0 = \Delta t^2 - \frac{2rv}{vv} \Delta t + \frac{rr - \sigma^2}{vv} \quad (2.1.6)$$

$$\Leftrightarrow \Delta t = -\frac{rv}{vv} \pm \sqrt{\left(\frac{rv}{vv}\right)^2 - \frac{rr - \sigma^2}{vv}} \quad (2.1.7)$$

But a caveat when executing on a floating point machine is present as can be seen when considering which solution is of interest. As for a possible collision it is necessary that the two particles move towards each other we can conclude that the scalar product is required to be negative  $rv < 0$ , because otherwise the particles are already moving away from each other.

Also the quadratic formula has two solutions, corresponding to the entry and the exit of the overlap. Because the entry has to be prior to the exit, we further conclude that interest lies on the smaller solution that is:

$$\Delta t = \frac{-rv - \sqrt{(rv)^2 - vv(rr - \sigma^2)}}{vv} \quad (2.1.8)$$

Now for the case where the distance of the spheres is already close to the diameter of the spheres we find  $(rv)^2 \gg (rr - \sigma^2)$ , which results in a cancelation of two large numbers leaving a small number. Floating point number operations are inherently bad suited because they tend to large inaccuracy in this case. Rewriting eq. 2.1.8 by making use of the third binomial formula **look if this is fine to write.** leads to:

$$\Delta t = \frac{(rr - \sigma^2)}{-rv + \sqrt{(rv)^2 - vv(rr - \sigma^2)}} \quad (2.1.9)$$

Comparably eq. 2.6.1 does not contain a cancelation of the type seen before and such is better suited for the use in a computer simulation. **cite Goldberg '91**

The event prediction algorithm proposed by Bannermann[2] works by differentiating 4 cases:

1. If  $rv > 0$  the particles move away from each other leading to a collision time of  $\Delta t = \infty$ .
2. If  $rr < \sigma^2$  an overlap is present resulting in an immediate collision time of  $\Delta t = 0$
3. If  $(rv)^2 - vv(rr - \sigma^2) \leq 0$  the two particles miss each other, including touching without momentum transfer, resulting in a collision time of  $\Delta t = \infty$
4. If none of the before is given the particles collide and  $\Delta t$  is calculated by eq. 2.6.1.

All collision times for a particle are then stored in a queue sorted by event time called particle event list (PEL). From the PEL the first entry is then passed to the global FEL.

This procedure initially takes place for all particles to set up the system and later on for particles involved in an event after its execution.

As will be discussed in section 2.1.2 some redundant calculations have been omitted in the actual simulation program, as well as a cell system added to reach  $\mathcal{O}(N)$  computation time, in turn introducing a new event type for cell changes.

A further detail to take care of is the possibility of scheduled events which have become invalid due to a earlier collision of the one of the particles. This is handled by assigning an interaction count to each particle and then store this at precalculation time with the event. When the event comes up, and the interaction count of one of the particles has increased in the meantime, the event is said to be invalidated. Depending on which particle had an event in the meantime the invalidation either causes no action or a recalculation of new events.

### 2.1.2 Details of the Implementation

As the simulation code is based on an earlier Monte Carlo Code for hard spheres a complete walk through the whole program would become quite extensive. Such we will focus on key points to understand the details of the simulation.

#### ***Event struct***

We start with the basic *Event* struct which includes 6 entries as shown in tab.2.1.1. The type of *time*

Datatype	Name of entry
(timeType)	time
(int)	event_type
(particle*)	particle
(void*)	partner
(int)	particle_count
(int)	partner_count

Table 2.1.1: Content of the *Event* struct.

(timeType) is usually set to double. The *time* variable itself represents the time for when the event is scheduled.

The *event\_type* variable is either set to 0 or 1 and indicates if the event is a cell transfer or a collision of two particles.

The *particle* variable is a pointer to the particle for which the event has been pre calculated, while the *partner* variable is set to be a void pointer. Such it is possible to either interpret it as a particle pointer for the collision type event or as an integer pointer to the index in the current cells' neighbours list for transfer events.

In the last two rows the interaction counts for particle and partner are listed as well. As the destination cell in a transfer event does not require an interaction count, the *partner\_count* variable is only used for collision events.

The *event* struct is used for all events throughout the simulation. For read and write operations with the HDF5 file format, the struct *event\_data* is available which uses only indexes instead of pointers.

#### ***Particle class***

The *Particle* class is comparably to the one from the MC code basis. Its MC related variables have been removed and additional key variables and concepts will be discussed in the following:

First a vector storing events called *backupEvent* has been added to make it possible to store events from the pre calculation for the case of the first event being invalidated. The idea of reusing events is discussed in many publications, for example that the memory cost increases only moderately with more backup events while the speedup does not increase much for more than two stored events [3]. It also has been argued that the added complexity can not account for the increase in efficiency[4]. Eventhough in the own simulations a decrease in calculation time of more than 10% was observed and the cost of complexity was seen as moderate. The difference might be explained by the fact that the systems under consideration in this thesis have a rather large particle density, leading to more invalid collisions.

In the context of reusing pre calculated results, it should also be mentioned that after a cell transfer the recalculation of events can be restricted to partner particles only in the new neighbouring cells, leading to only 1/3 of the calculation time in this case. But as mentioned systems under consideration are mostly rather dense and such the number of transfers is often at below 5%. Thus the increase in efficiency was assumed to be too costly on the complexity side, and not implemented. Eventhough for sparse systems, it might make sense to include an *updatePEl* routine.

Also key differences to the former MC Particle type are the variables *total\_interactions* and *particle\_delayed\_time*. The first is the variable for book keeping of interactions, while the second represents the event driven character of the simulation. Because each particle only moves on purely ballistic trajectories until an event occurs, it is not necessary to keep all particle positions and velocities synchronized in time. Quite on the contrary it would mean executing extra operations together with summing rounding errors by each floating point operation.

Because sometimes it is desired to have the whole configuration at one point of time, the *transferToTime()* function of the particle provides the possibility to take the particle into the present. This is necessary soon as measurements are performed on the system, including snapshots.

As mentioned before the system behaves chaotic even under slightest changes like a rounding error from an extra floating point operation. A result of this is that measuring at different rates during a simulation changes the simulation trajectory quite a bit. It has been observed that such a system may keep close to the undisturbed trajectory for about 50-100 events/particle. As it is of desire to measure quantities and take snapshots without disturbing the simulation, the simulation program makes employs copies of the configuration being costly in terms of memory but making simulation resets or higher sampling rates at interesting points possible within a defined trajectory.

The structure is as follows: The first copy is rewritten with an image of the working configuration just



before any measurement. The working trajectory itself is then disturbed by the measurement, and afterwards replaced with its state before the measurement from the backup configuration.

The second copy actually includes the full simulation state, while the first only includes the particle configuration. This second one might be used to save a state during the simulation and reset to just the same point at any later time.

### The *Box* class

The box of the simulation stayed mostly the same as in the previous MC code. One change is the array *neighbours\_lookup* which has been added. It contains the indices for the cells' *neighbours* array pointing to cells that share their surface. It is used to identify which cell a particle has to be transferred to during a cell transfer event.

Furthermore the *Update* routine now takes care of all quantities depending on the length of the box, making the *rescale* routine a simple rescaling of the edge lengths with an additional *Update* command.

### The *Scheduler* class

While the afore mentioned parts of the program are necessary for the EDMD program, the *Scheduler* class contains the most distinct parts of the program. It keeps track of all events to come, predicts new events and orchestrates the execution of the events. The essential functions are discussed in the following subsections while some basic properties are shortly highlighted here.

First of all the *Scheduler* holds the Future event list (FEL) in which at least one event per particle is stored. As discussed within section 2.1.2 the simulation is capable of saving the complete state of a trajectory, including all pre calculated events. For this purpose an array of *Events* is available.

Furthermore the *Scheduler* includes the *global\_time*.

Important for the efficiency is the pre allocation of all arrays used within the prediction calculations, as the number of executions for the collision prediction routine is about  $\frac{30}{\text{particle} \cdot \text{step}}$  accounting to a few billion function calls during a small simulation.

#### *Scheduler::predictTransfer()*

As the name suggests this function predicts the next cell transfer of a particle due to its movement. For this it calculates the position of the particle at global time, which for a valid state of the simulation always lies within its cell. By transforming the momentary position of the particle from the global

coordinate system to the coordinate system of the cell and taking into account the periodic boundary conditions, we can write for each dimension  $i$  the equations

$$t_{i1} = -\frac{r_i}{v_i} \quad \text{and} \quad t_{i2} = \frac{l_i - r_i}{v_i} \quad (2.1.10)$$

which describe the times when the particle pierces the cell's left and right boundaries. A negative time corresponds in this case to a boundary crossing in the past, a time comparable to 0 means that the particle is on the edge of its cell and a positive time means that the boundary crossing lies in the future. By going through the different possible cases for  $t_1$  and  $t_2$  we find the resulting next crossing time for each case as shown in tab. 2.1.2.

$t_1$	$t_2$	Result	Case
>	>	invalid	-
>	=	$t_{\text{crossing}} = t_1$	0
>	<	$t_{\text{crossing}} = t_1$	1
=	>	$t_{\text{crossing}} = t_2$	2
=	=	invalid	-
=	<	$t_{\text{crossing}} = t_1$	3
<	>	$t_{\text{crossing}} = t_2$	4
<	=	$t_{\text{crossing}} = t_2$	5
<	<	invalid	-

Table 2.1.2: Possible results for left and right crossing time with resulting choice of next crossing time. >, = and < are to be read as for example  $t_1 > 0$ . The case indicates the case number within the actual simulation.

By collecting the next crossing times for each dimension and taking the minimum of these times the exit time of the particle from its cell is determined.

The return value of the routine is an *Event* where the partner is given as an address to the box' *neighbours\_lookup*. The index lies between 0 and 5, corresponding to the 6 possible neighbour cells sharing a surface with the current cell of the particle. Each valid case represents a distinct neighbour cell and the index within the cells *neighbours* array is clearly defined by the cell setup routines and is shown in tab. 2.1.3.

dimension	boundary	case	index
x	front	0	12
	back	1	13
y	front	2	10
	back	3	15
z	front	4	4
	back	5	21

Table 2.1.3: Overview of the cells' *neighbours* indices directly sharing a surface for 3 dimensions. As the indices hardly follow any simple pattern they are explicitly noted at this point. Obviously the cell consists of a front and a back boundary in each dimension. The corresponding case matches the one from tab. 2.1.2.

### ***Scheduler::predictCollision()***

The prediction of collision times has already been discussed in section 2.1.1. The implementation in the program first calculates all necessary scalar products while accounting for the periodic boundary conditions, and in a second step returns the collision time depending on the case at hand.

The presented algorithm is only valid for single sized particles. If polydisperse systems are supposed to be considered the algorithm has to be adjusted. mayhap do it in the appendix?

As this routine is executed through out the simulation very often it has been tried to optimize its efficiency multiple times. For example calculating only necessary results for the next case differentiation has been implemented but no significant increase in efficiency was recognized and for better readability the prior version has been reestablished. In either case if an optimized way of calculating the results is found it might be useful to use them. ← Not really nice

### ***Scheduler::setupFEL()***

This routine fill the FEL of the simulation. For this purpose it iterates through all particles and calls *setupPEL* for each of them. The PEL in turn is set up by predicting the next cell transfer as well as the next collisions with all particles within the  $3^d$  cells directly surrounding the particle. From all predicted events only such with finite times are then written to the *backupEvents* vector corresponding to the PEL of the particle.

For the FEL only the top event of each particle is then used. Because other events from the PEL are able to move on to the FEL ounce written to the FEL an event has to be erased from the PEL.

***Scheduler::executeTransfer()***

The execution of a transfer event is accomplished by the event particles *MoveBetweenCells()* routine. The departure cell is taken as the event particles own cell. While the event partner holds the information which of the cell neighbours is the destination cell.

***Scheduler::executeCollision()***

The outcome of a collision between particle 1 and 2 with corresponding position and velocity can be derived by momentum and energy conservation cite some textbook, or better jsut write down the calculation too be

$$\vec{v}'_1 = \vec{v}_1 + \left( \frac{\vec{r} \cdot \vec{v}}{\vec{r} \cdot \vec{r}} \right) \vec{r}_{12} \quad (2.1.11)$$

While eq. 2.1.11 is not directly depending on the radius, an indirect dependence by the collision time and configuration at which eq. 2.1.11 is evaluated persists.

***Scheduler::executeEvent()***

The execution of an event works in multiple steps. At first the topmost *Event* is copied from the FEL where it is deleted. Next the validity of the interaction counts of both particle (*cond1*) and its partner (*cond2*) are evaluated. The validation is nothing else than a comparison of the interaction counts when the event was scheduled with the present interaction counts. As the conditions are used in the following flow statements they are stored as boolean type. Furthermore in the case of a transfer event the validation of the partner is not necessary but only for better readability performed.

It follows a distinction between 5 cases which are given by:

**Valid transfer (*event\_type*==0 and *cond1*)**

The transfer is executed, the global time is evolved to the event time and the particles PEL is rebuilt and its next event pushed to the FEL.

**Valid collision (*event\_type*==1 and *cond1* and *cond2*)**

The collision is executed, the global time is evolved to the event time and for both participating particles new PEL's are built and each top event is pushed to the FEL.

**Invalid transfer (*event\_type*==0 and not *cond1*)**

The particle must have had an interaction previously where a new event for it was scheduled, thus no action is taken.

**Invalid collision due to particle (*event\_type*==1 and not *cond1*)**

The particle must have had an interaction previously where a new event for it was scheduled,

thus no action is taken as for the above.

### **Invalid collision due to partner (*event\_type*==1 and not *cond2*)**

Only the partner had an interaction previously where a new event for it was scheduled, thus a new event for the particle is required. As the particle had no further interactions, the *backupEvents* are still valid and its first entry is pushed into the FEL. In case no backupEvents are stored the PEL is rebuilt and its first entry pushed to the FEL instead.

The order of the cases might be exchanged, except for the last two. This is because the last one assumes *cond1* to be true, which is guaranteed by the case before.

Furthermore the routine counts the number of each case, to monitor numbers of collisions, transfers and invalidated cases by type. This is not required by the simulation but can be helpful for understanding the system and simulation.

## **2.2 Measurement quantities**

list quantities and their use etc.

### **2.2.1 Mean squared displacement**

- Averaged or as distribution
- Required for diffusion

### **2.2.2 Radial distribution function**

- Well show an RDF?

### **2.2.3 Largest Cluster**

- Non local
- large in case of nucleation
- requires almost no memory

### **2.2.4 Cluster size distribution**

- Non local
- requires more memory still not as much as for example particle positions

-

### 2.2.5 Cluster positions and numbers

- not done
- local as single clusters can be named and followed from formation to melting or final growth
- Much more information as locality can be achieved.

### 2.2.6 Tensor of Gyration

- Only done for largest cluster of size larger than some threshold
- How to extract the eigenvalues l1,l2,l3
- Radius of Gyration
- Anisotropy
- Asphericity

## 2.3 Probe of simulation code

To probe the simulation dynamics we measured the longtime diffusion constant and the radial distribution function of the stable hard sphere liquid, as there are many measurements available in the literature to compare with.

### 2.3.1 Diffusive behaviour

The diffusive behaviour of particles in a liquid usually can be separated in three distinct parts. First the short time diffusion which can be understood as the random movement of the particles within their momentary cage within the fluid. Second a sub diffusive phase in which the particles are repelled for the first time by their nearest neighbours. And third the long time diffusion to describe the random propagation of the particle through the fluid over time.

As the ballistic hard sphere system enters into the long time diffusion almost at once only this is really measurable. By assuming a diffusive process we have the expectation that the average mean squared displacement (MSD) of a particle can be well described by

$$\langle x^2 \rangle(t) = 2 d D t, \quad (2.3.1)$$

where  $\langle x^2 \rangle$  is the expectation value of the MSD,  $d$  the number of spatial dimensions,  $D$  the characteristic diffusion constant, and  $t$  the time by which the system has evolved.

By measuring  $\langle x^2 \rangle(t)$  and making a linear regression to the data points we can find the Diffusion constant  $D$ .

To probe the simulation code, systems as characterized in tab.2.3.1 have been used. The equilibration phase has been carried out up to  $\eta_f = 50\%$  at the final volume fraction while above  $\eta_f = 50\%$  an initial volume fraction of  $\eta_i = 45\%$  has been used to obtain a fluid rather than a solid after the equilibration phase. As the measurement stretches into the metastable regime, it has also been necessary to check for nucleation events which were not present during the measurements.

The resulting diffusion constants depending on the fluid volume fraction are shown in fig.2.3.1 alongside values obtained from the literature, for a similar system.

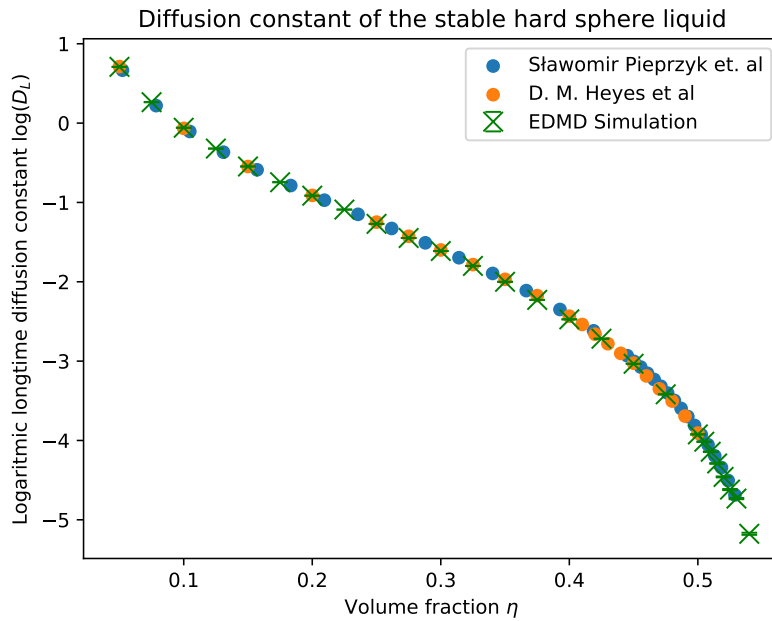


Figure 2.3.1: Logarithmic long time diffusion constant of the hard sphere liquid as measured in the own simulations as well as measurements from the literature. *cite the people.*

As it can be seen the EDMD simulation is very well capable of reproducing the diffusion constant for the hard sphere liquid, and such we expect the dynamics of it to accurately represent the purely ballistic hard sphere system.

Parameter	Value
N	16384
eq_steps/particle	5000
pr_steps/particle	20000
$\eta_i$	5% ... 50 %
$\eta_f$	5% ... 54 %

Table 2.3.1: Input parameters of diffusion test systems.

### 2.3.2 Radial distribution function

A further well known quantity for the hard sphere system is the radial distribution function. As a theoretical prediction the Percus-Yevick approximation can be used to compare with, also it would be possible to compare with Monte Carlo simulations of the hard sphere system. In fig. 2.3.2 an overview for a range of volume fractions is shown from the same simulations used in section 2.3.1. Clearly visible is that no particles enter within the diameter of the spheres. Further for higher volume fractions the liquid shells become very well visible. At very high volume fraction we also find that new peak arises below  $r = 2\sigma$ .

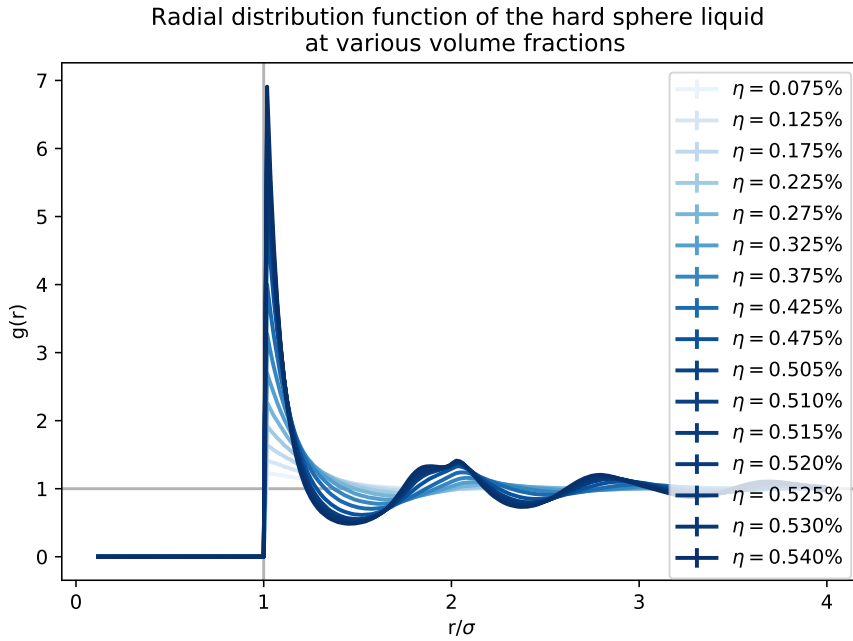


Figure 2.3.2: Radial distribution functions for a range of volume fractions. The colouring corresponds to the used volume fraction.

To compare with Percus-Yevick approximation the radial distribution function for two single volume fractions is shown with the corresponding theoretical solution in fig. 2.3.3.



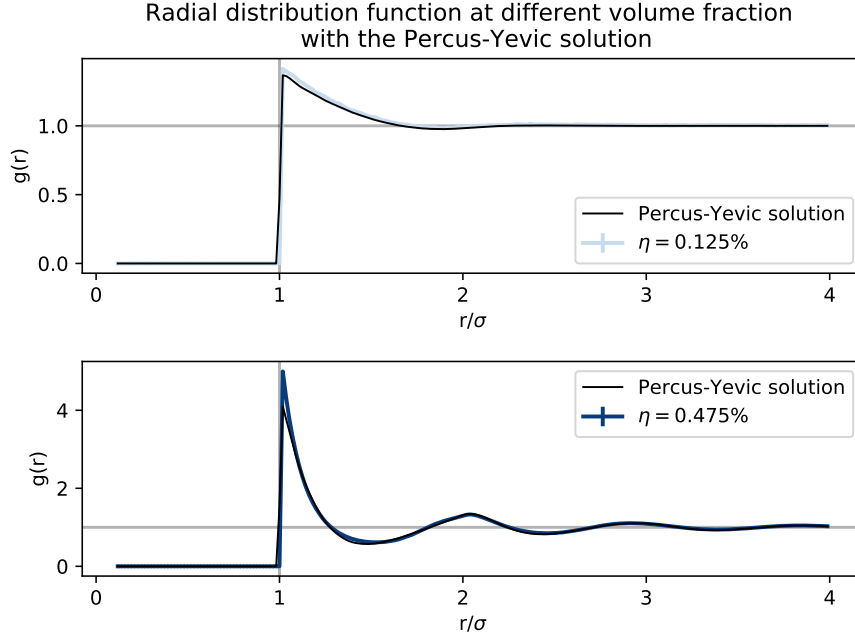


Figure 2.3.3: Radial distribution function for the hard sphere system at a low and at a high volume fraction of the liquid together with the theoretical prediction from the Percus-Yevick approximation.

As highlighted for example in [Hansen2006] the theoretical approximation has some flaws as can be seen with  $g(r)|_{r=1\sigma}$  being too low for the Percus-Yevick approximation. Eventhough we see that overall the two radial distribution functions follow each other rather closely.

Such we are confident that the elaborated simulation code is capable of producing accurate data in other contexts as well.

It seems in the hansen and mcdonal that the peak below 2 sigma is also not present in their MC  $g(r)$ , is there a reason for this?

## 2.4 Estimate of required resources

To choose system parameters senseful calculation times and file sizes of the simulation were characterized. This was of interest as the program was supposed to run on the NEMO high performance computing cluster which puts hard boundaries on calculation times which when treepassed can cause tremendous loss of data if not corretly caught by the program.

### 2.4.1 Calculation time estimates

The calculation time of the program was tested for a large range of different system sizes up to almost 9 million particles in the fluid state. As can be seen in fig. 2.4.1 the calculation time increases proportional to the system size for the execution of a step as well as for a measurement of the fluid system. The calculation cost being of  $\mathcal{O}(N)$  enables the study of large systems. Furthermore from the slope an expectation for the execution time of a single event can be deduced, as well as an expectation for the time necessary for a measurement. As discussed on the example of fig. 2.4.2 the dependence of the measurement routines on the largest cluster size were not seen here, as possible clusters remained rather small during these simulation times.

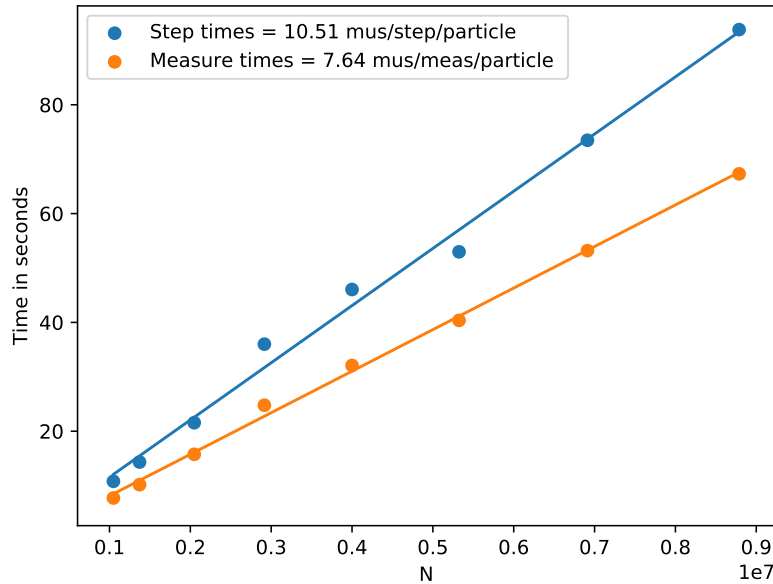


Figure 2.4.1: Overview of CPU time required for calculating a simulation step, consisting of an event for each particle, and a measurement of relevant quantities for the system. As assumed for a simulation algorithm with  $\mathcal{O}(N)$  calculation effort, the data points can be described by a line rather well. As the CPU time is clearly related to the further workload of the CPU during the calculation it is also expected to find fluctuations if the other workload of the machine is not strictly controlled.

The effect of larger clusters was only investigated after problems with the runtime of the programs were traced back to these. The q6q6-order parameter routine was tested for larger clusters in a nucleating simulation with about 1 million particles within the box. As can be seen in fig. 2.4.2 the calculation cost of the cluster finding routine can be described with a quadratic dependence on the largest cluster. For an impression what this means we can use the calculation costs of a simulation

step from fig. 2.4.1 being about  $t_{step} \approx 10 \mu s/\text{particle}$ . Such the execution of one step takes about 10s for 1 million particles. If a measurement is performed every 10th step, the calculation cost of the measurements without a large cluster remain below 10%. But as the largest cluster grows to a few hundred thousand particles in size, the measurements can make up 30 % and more of the calculation cost, or for a fixed number of steps, increase the calculation time by about 50 %. This previously unseen effect lead to actual data loss as the combination of NEMO cluster policy and EDMD simulation program did not result in a save shutdown of the program after breaching the walltime limit of the NEMO cluster.

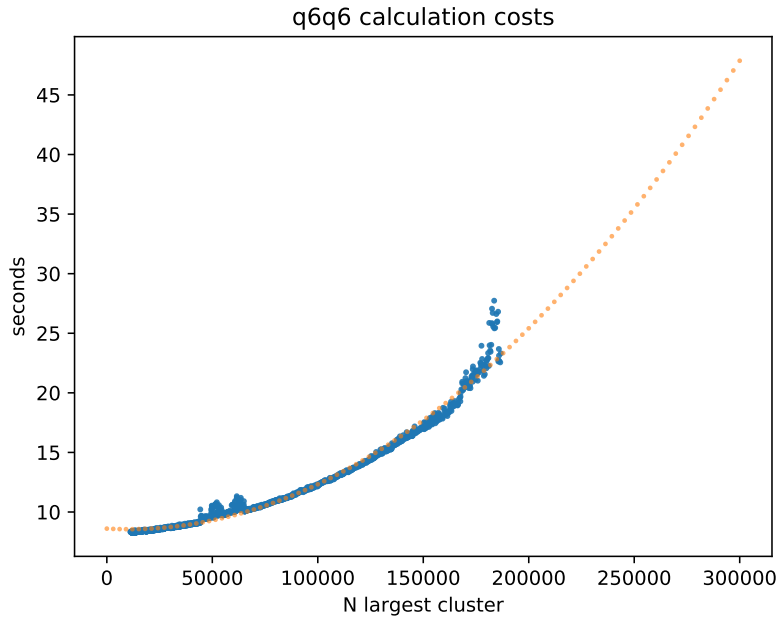


Figure 2.4.2: Calculation time of the q6q6 order parameter at an increasing largest cluster size during one nucleation, together with the quadratic best fit indicating that the q6q6 routine calculation effort can be approximated by  $\mathcal{O}(N_{lc}^2)$  where  $N_{lc}$  is the size of the largest cluster.

### 2.4.2 File sizes estimates

A further important constraint for the simulations are the produced amount of data. To get an impression of the filesizes, the required memory for snapshots, reset steps and other measurements were measured prior to the actual simulations. The results for a single snapshot containing all positions and velocities of all particles as well as the size of a single simulation reset step containing all positions, velocities, the FEL, all PEL's and all delayed times is shown in fig. 2.4.3. It can be seen that the file

size is directly proportional to the system size which clearly expected as each particle adds a further set of positions, velocities etc. to the saved data.

The memory costs of other measurements have been left out of fig. 2.4.3 as these only amount to substantial filesizes if measurements at about each step for long simulations are done.

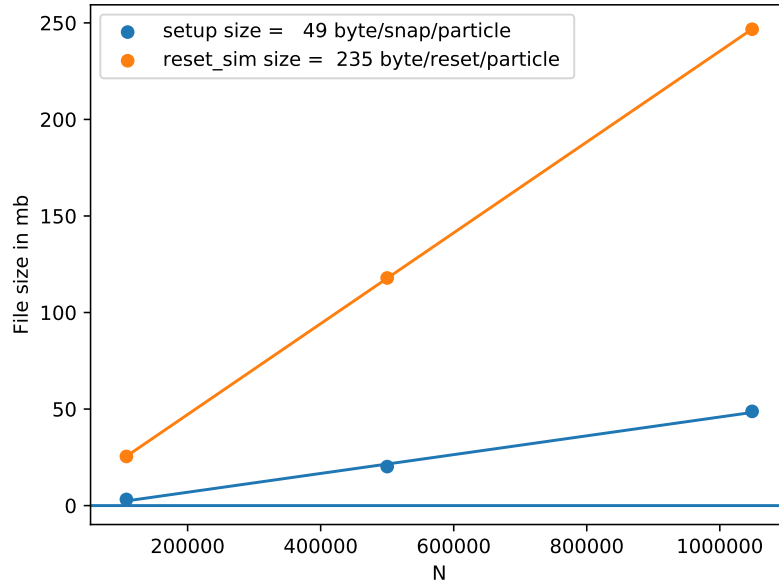


Figure 2.4.3: Overview of filesizes when a single setup on the one hand and a single full simulation on the other hand is saved for comparison reasons together with their corresponding linear regression. while the linear regression for 3 points is statistically not exceedingly senseful it still remains a useful tool to extract the slope which corresponds to the required memory per particle and snapshot or reset simulation.

## 2.5 First produced data

The motivation for the simulation code is based on the interest in nucleation rates of the hard sphere system at varying volume fractions. To observe a nucleation the volume fraction of hard spheres has to be changed rapidly from lower ones where the system is in the stable fluid phase to higher ones where a meta stable fluid-solid phase exists. If this metastable phase is evolved in time nucleations can be observed as stochastic distributed events. To measure those without effects originating from the handling of the simulation, some parameters were tested within reasonable ranges prior to the data production.

For this simulation the equilibration steps as well as the initial density before the volume quench

Parameter	Value
N	16384
eq_steps/particle	100 ... 20000
$\eta_i$	5% ... 49 %
$\eta_f$	54 %

Table 2.5.1: Input parameters of test systems probing the dependence on equilibration steps and initial density.

seemed like they could introduce unwanted artefacts, and thus we performed some smaller data series to evaluate if and when these effects might come into play.

The used test system is characterized by the figures in tab. 2.5.1.

The general behaviour of the systems is analysed by inspecting the cluster distribution over time. The mean cluster distribution is shown in fig. 2.5.1 together with the same data smoothed by a gaussian filter matrix. The smoothing is used because in a next step the difference between the mean cluster distribution and the cluster distributions with varying simulation parameters is compared, and without smoothing at low count rates only fluctuations are visible.

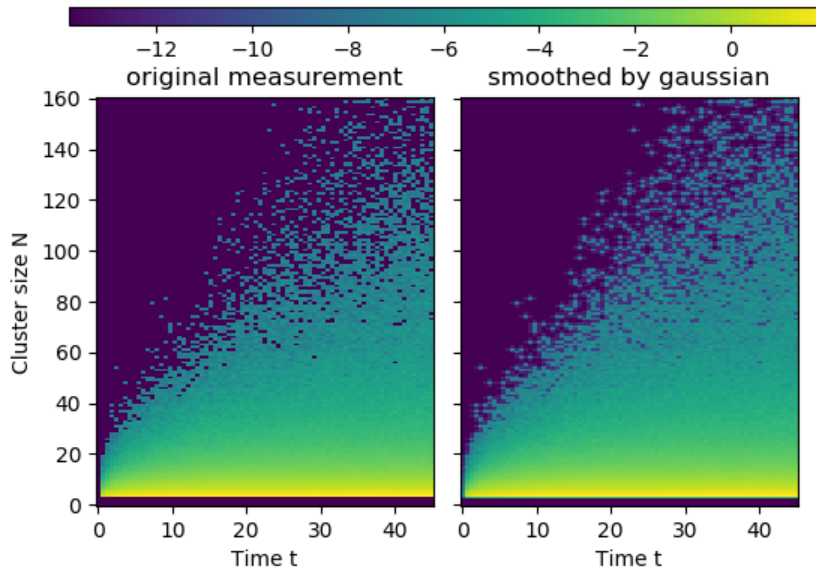


Figure 2.5.1: Heat map of the mean cluster distribution over time. The diagram encompasses 800 trajectories of 16384 particles each. The colouring indicates the logarithm of the mean cluster occurrence corresponding to a probability in the stationary case.

From fig. 2.5.1 we can see how the system behaves after a volume quench into the metastable region. In the liquid rarely any clusters are present and thus directly after the quench no clusters are present

either as the spatial configuration requires time to rearrange into clusters. In the later evolution we see how clusters form, and soon after begin to nucleate leaving the range of the diagram.

To compare simulations with varying parameters the quantity defined in eq. 2.5.1 is used, where complications with zero values are circumvented by fixing these values below the regular signal.

Three samples of this comparison are shown in fig. 2.5.2 and fig. 2.5.2. The colouring indicates  $\Delta_{p(N,t)}$  defined in eq. 2.5.1. As mentioned above the quantities  $p_i(N,t)$  and  $\langle p(N,t) \rangle$  have been smoothed by a gaussian filter, because the number of samples included, with 100 trajectories per series, were not sufficient to produce smooth distributions at the given sampling rate. Such without smoothing only fluctuations would be visible.

$$\Delta_{p(N,t)} = \log\left(\left|\frac{p_i(N,t)}{\langle p(N,t) \rangle} - 1\right|\right) \quad (2.5.1)$$

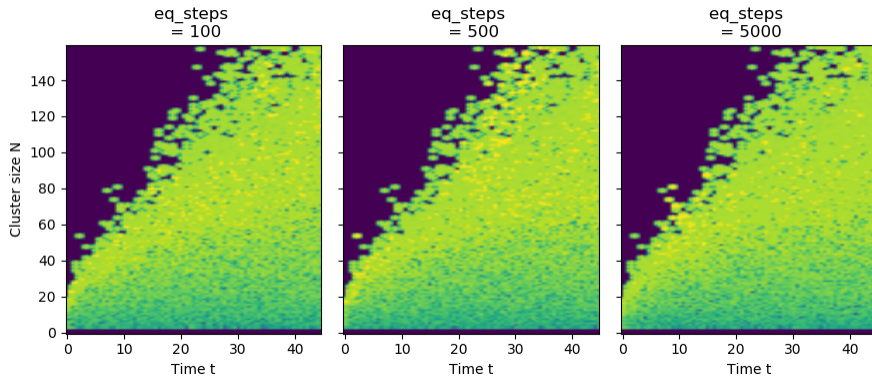


Figure 2.5.2: Heat map of differences between the cluster distributions within simulations carried out with varying the length of the equilibration phase. The quantity used for colouring is defined in eq. 2.5.1, where yellow indicates a large difference while blue indicates a small difference. Providing a legend of the colouring is omitted as  $\Delta_{p(N,t)}$  has no further use as to indicate differences and actual values do not add any use.

On first sight none of them differ in their general behaviour. Because at  $t=0$  after the quench no clusters have formed yet and also no clusters were present in the stable liquid, the difference between all simulations is zero, indicated by the blue region in the top left corner. The features visible on the edge between the zero region and the nonzero region on the other side are the same, because they are features of the mean distribution carried through. Actual differences not due to fluctuations can only be seen within the green and yellow non-zero region, but none such differences are observed.

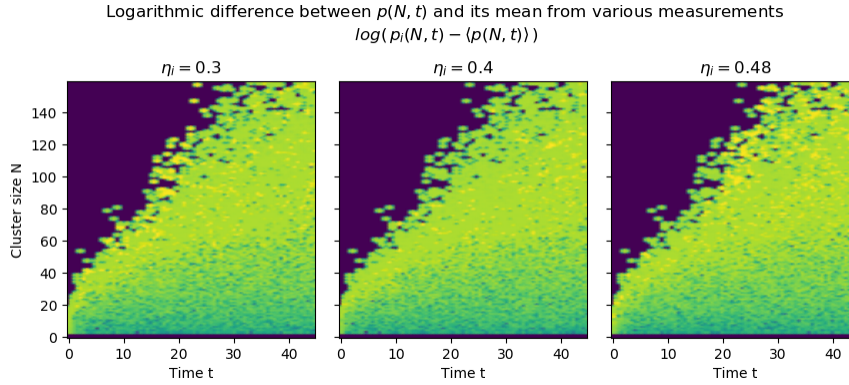


Figure 2.5.3: Heat map of differences between the cluster distributions within simulations carried out with varying the volume fraction of the liquid during the equilibration phase. The quantity used for colouring is defined in eq.2.5.1, where yellow indicates a large difference while blue indicates a small difference. Providing a legend of the colouring is omitted as  $\Delta_{p(N,t)}$  has no further use as to indicate differences and actual values do not add any use.

While it seems like the initial volume fraction of  $\eta = 0.4$  and  $eq\_steps = 5000$  include less irregular fluctuations, dramatic effects from choosing the simulation parameters can be excluded. Interesting in this context are especially the simulations with  $eq\_steps = 100$  because after executing 100 events/particle on average, the initial perfect crystal configuration is only on the verge of not being detected anymore. Such one could expect that these configurations might be still close to crystallization, but instead we do not detect any significant difference.

A further more quantifying analysis of the differences is given by calculating the mean nucleation rates assuming classical nucleation theory. This is done for the data shown in fig.2.5.4. The calculations of the rates have been carried out as described in section 3.5.

As we see no significant difference in the nucleation rates can be observed even for the bold setting of only 100 events per particle for the equilibration phase. Eventhough it can be seen that for this special case the rate is a little higher, but it may as well be a stochastic fluctuation.

Overall we conclude in this chapter that as long as parameters are set within reasonable boundaries, we expect not to have systematic influences of simulation parameters.

## 2.6 Possible extensions

The program at this state is capable of simulating large systems including compression and relaxation. Extensions can aim at either increasing complexity or size of the systems.

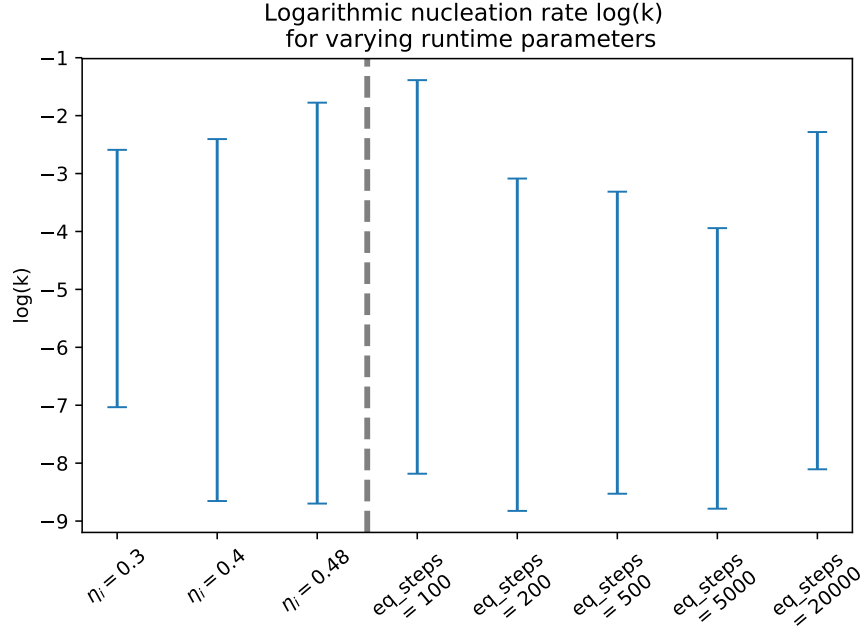


Figure 2.5.4: Comparison of nucleation rates under CNT assumptions for different initial densities during equilibration with eq\_steps fixed at 5000, as well as varying eq\_steps with  $\eta_i$  fixed at 0.45.

### 2.6.1 Varying radius

A further notch of complexity would be including polydispersity into the simulation. For this purpose the prediction of collisions has to be adjusted. When looking at the derivation of eq. 2.6.1 it is found that  $\sigma$  has to be changed to  $R_i + R_j$

$$\Delta t = \frac{(rr - \sigma^2)}{-rv + \sqrt{(rv)^2 - vv(rr - \sigma^2)}} \quad (2.6.1)$$

### 2.6.2 Varying mass

Same as with radius, highlight the points in the code requiring a change.

### 2.6.3 Multiprocessing

Idea on Parallelizing the code. Probably more advanced, but for larger systems (Really large files) doable. (Actually RAM might become a serious problem).

Besenrein, Samstag 6.3. mittag, 13 h



## 3 Data Analysis

This is the analysis part

### 3.1 Cluster growth

Cluster growth depending on density

### 3.2 Tensor of Gyration properties

Well only swamp here, but it can be shown to conclude the swamp. - theory

### 3.3 ACF largest cluster?

Just in case anything can be seen here

### 3.4 Nucleation time dilemma

Evaluation of induction time. Problem with accuracy and precision. Compare methods.

- General induction time ideas.

-(HaJo : induction time of crystal or induction time of precursor under the assumption of later growth.)

#### 3.4.1 induction time by critical cluster from cnt

-r\_crit calc and induction time distribution

#### 3.4.2 induction time by constant growth extrapolated to zero

-linear regression and distribution of constant growth rates

-induction time distribution

### 3.4.3 induction time by exponential fit

- maybe procedure
- distribution rates
- distribution induction times

### 3.4.4 induction time by fluctation horizon crossing

- non locality of fluctuations -> acf?
- sensefull to use the data until it vanishes
- induction time distribution
- discuss the delay distribution (precursor structuring time)
- evaluation of an simple induction time or rate by cnt next chapter

## 3.5 Induction time by exponential distribution

Obtain exp assumption and best estimator

- CNT constant nucleation rate(justified by fast nucleations depending on the delay)
- For such expontneila ML estimator following statistics papers.
- Uncertainty by MC method following numerical recipes.
- 

## 3.6 Nucleation rate comparison

All Nucleation rates that can be found.-> mayhap ask Hajo.

- nucleation rates without
- nucleation rates with small particles

## 3.7 Memory Kernels

Memory kernels of systems at various densities. Depends strongly on what is found here -memory kernels of 16k system at varying points of time -maybe memory kernels of 1m system, but do not know what to say. Maybe only mention that after seeing the 16k system to have only memory kernels at about middle of the time, no true memory kernel is visible in the 1m system as volume fractions have been too low with too large calculation times.

## 4 Conclusion - Summary

### 4.1 Conclusion

## 5 Appendix

### .1 A

# Bibliography

- <sup>1</sup>D. M. Heyes, M. J. Cass, J. G. Powles, and W. A. Evans, “Self-diffusion coefficient of the hard-sphere fluid: System size dependence and empirical correlations”, *Journal of Physical Chemistry B* **111**, 1455–1464 (2007).
- <sup>2</sup>M. N. Bannerman, S. Strobl, A. Formella, and T. Pöschel, “Stable algorithm for event detection in event-driven particle dynamics”, *Computational Particle Mechanics* **1**, 191–198 (2014).
- <sup>3</sup>M. N. Bannerman, R. Sargant, and L. Lue, “DynamO: A free  $O(N)$  general event-driven molecular dynamics simulator”, *Journal of Computational Chemistry* **32**, 3329–3338 (2011).
- <sup>4</sup>A. DONEV, S. TORQUATO, and F. STILLINGER, “Neighbor list collision-driven molecular dynamics simulation for nonspherical hard particles.II. Applications to ellipses and ellipsoids”, *Journal of Computational Physics* **202**, 765–793 (2005).