Particle-based Simulations (6KM59)

Dr. Ir. J.T. Padding

April 16, 2013

CONTENTS

PREFACE

Particle-based computer simulations play an increasingly important role in science and technology, for reasons that I discuss in the first chapter. Since the 1950's a large number of methods have been developed to tackle all sorts of problems, ranging from the motion of molecules to the motion of colloidal particles, sand particles, rocks, boulders, planets, and even galaxies. The methods go by many names such as molecular dynamics, Langevin dynamics, Brownian dynamics, dissipative particle dynamics, stochastic rotation dynamics, discrete element model, discrete particle model, etcetera. For the uninitiated, the sheer *number* of different methods must look overwhelming.

However, what I have learned over the years – starting from my graduation work in 1997 on molecular dynamics of polymer melts to my current work on granular systems, fluidized beds, spray dryers, etcetera – is that in their core there is a *great deal of similarity* between all these particle-based methods. Having learned the general principles and tricks of the trade at one level, it is relatively easy to apply them at another level. Thus, in my opinion, it is important to know about common elements of (almost) all particle-based simulation methods. This is what I aim to do in chapter 2. Of course there are peculiarities that pop up only at certain scales, which is why I spend the second half these lectures on microscale, mesoscale and macroscale simulations.

These lectures would not have been possible without my intense interactions with four people. First and foremost, I am indebted to my PhD tutor, Prof. Wim Briels (University of Twente), who has introduced me to the world of molecular dynamics, coarse-graining, and learned me to appreciate both the subtle points and the strength of statistical mechanics. Second, I am indebted to Dr. Ard Louis (University of Cambridge / Oxford University), with whom I have had many wonderful adventures into the low-Reynolds number world of colloidal dynamics and stochastic rotation dynamics. Finally, I am indebted to Prof. Hans Kuipers and Dr. Niels Deen (Eindhoven University of Technology) for accepting me in the world of macroscopic particle simulations combined with fluid mechanics, and teaching me the subtle points of the world at higher Reynolds numbers.

Finally, I would like to thank Luuk Seelen for throroughly proof-reading these lecture notes. Still, because these notes have been produced almost entirely during just two frantic months in March and April 2013, I am convinced that some errors will remain. Please report them to me, and I will update them for the next edition.

JTP

The Hague, 15th April 2013.

CHAPTER 1

GOALS OF PARTICULATE MODELING AND SIMULATION

1.1 Chapter objectives

Through the course of this chapter, you will accomplish the following:

- You will learn about different goals of particle-based simulations.
- You will learn about simulations as a third branch of science.
- You will encounter a first example of a particle-based simulation, namely a molecular dynamics simulation of neutral spherical molecules.

1.2 Simulation: a third branch of science

Some texts on computer simulations start with an overview of current computing capabilities, boasting about teraflop processing speeds and terabyte memory capacities. I will not give in to this temptation because it is not my intention to make these lecture notes look horribly outdated in five to ten years. The core message is that with the ever increasing computational power available to us, increasingly large and complex systems can be modeled and simulated on a computer.

These lectures focus on a particular type of computer simulation, *particle-based* simulations, where particles interact with each other and with external fields. You will learn *how* to program and perform such particle-based simulations, with a focus on fluid and fluid-solid systems such as molecular fluids, suspensions of colloidal particles, and fluidized granular particles. But before delving into this, the first question

that needs to be answered is *why* one would be interested in performing such simulations anyway? It may be argued that science has progressed just fine for hundreds of years by careful experimentation, summarising observations in laws of nature, theoretical derivations of the consequences of these laws, confirmation by experiments, etcetera, without the use of any simulation.

A simple answer to the above question is that the trajectories of a large group of particles quickly become too difficult to solve analytically. But this is also a boring answer. The more interesting answers come when we consider the different *goals* of simulations. The following list is not exhaustive, but gives a flavour of the different goals of simulations, with some examples.

- Often the goal of a simulation is to *study the collective behaviour* of the countless interactions between the components of a system. It is an amazing fact that even very simple interaction rules (forces) between very large numbers can lead to very complex emergent behaviour. For example, simple collisions between the molecules in a liquid and between the molecules and sand particles lead to fascinating swirly patterns in the sedimentation of sand through water.

- A deeper goal of a simulation is to *generate a fundamental understanding* or an *explanation* of the phenomena or processes occurring in real or model systems. In the simulation, you have full control over the objects, interactions and forces. This allows you to find the determining factors in a phenomenon or process. For example, you may ask whether the mutual uncrossability of polymer chains has an effect on the diffusion. By simulating polymer chains with and without such a constraint, you can discover that there is hardly an influence for relatively short polymers, but a very large effect on longer polymers.

- Sometimes the goal is to obtain information about a system which is *difficult to access or control experimentally*. With simulations we can perturb and do computational measurements in systems which are too small to observe accurately (molecular processes on sub-nanometer scales), too large to influence (dynamics of galaxies of stars), or difficult to achieve experimentally (very high temperatures or pressures).

- Another goal of simulation is to *predict* the behaviour of a system. From an engineering point of view, this enables us to optimise a process, or make a choice between different designs. For example, rotating impellers are often used in container tanks to keep the contents well mixed and baffles are used to stop the swirl. There is much freedom in the number, shape and size of these impellers and baffles. Simulations enable us to explore a large number of designs, and to select the best candidates for further experimental testing.

The above list shows why simulation has appeared as a third main branch of science, situated between experimentation and theory. Simulations enable us to study theoretical models which are simply too difficult to solve analytically, or systems which

are too difficult or expensive to access experimentally. Simulations allow us to play with the system in a virtual world in which the simulator has full control over all elements, and thus generate explanations for experimental observations.

1.3 Validation of a model

Before you can play god of your own virtual world, a word of warning is at place:

> "Garbage in → garbage out"

The above quote means that the outcome of a simulation is only as good as the assumptions that go into the model. Unless the simulation is meant to explore the consequences of a well-defined theoretical (toy-)model, it is important to check the validity of the simulation model against experimental results.[1] In other words, the simulation should reproduce with good accuracy some of the observed quantities in well-controlled experiments of the same system. Only then the model can be used to generate understanding, perform numerical measurements, or predict the behaviour in other circumstances. The definition of "good accuracy" depends on the precise goal of the simulation and the quantities one is interested in, and therefore differs from case to case.

1.4 Practicum: phase behaviour of spherical molecules

As an introduction to the field of particle-based simulations, you will study the phase behaviour of molecules though an interactive Molecular Dynamics simulation java applet.

Molecules can organise themselves in different phases, such as a solid, a liquid or a gas. The phase of a material depends on the chemical structure of the material, the temperature and the pressure or density. At high enough temperatures the distinction between a liquid and a gas disappears, and we speak of a fluid. In this practicum you will study the phase behaviour of neutral sperical molecules such as argon, krypton (which are actually atoms) or methane (which is approximately spherical). You will investigate if or how intermolecular interactions, molecular mass, and temperature affect the formation of a phase.

Answer the questions in the html page "explorephaseNew.html" and allow the java applet to run on your computer. Discuss the questions and answers with your neighbour.

[1] By validity we do *not* mean a correct implementation of the model. A correct implementation must be checked by performing simulations for analytically calculable special cases. This may be called *verification* of the (implementation of the) model. However, the assumptions of a special case probably include all assumptions of the model, and therefore do not validate the model.

CHAPTER 2

GENERAL PRINCIPLES OF PARTICLE-BASED SIMULATIONS

2.1 Chapter objectives

Through the course of this chapter, you will accomplish the following:

- You will learn about similaries in structure of (almost) all particle-based simulations.
- You will learn to distinguish different types of forces such as conservative versus dissipative forces, and external forces versus multi-body forces.
- You will learn to calculate pairwise forces in a simulation code.
- You will learn to use neighbourlists and cell-linked-lists to speed up simulations.
- You will learn to apply different types of boundary conditions, including walls and periodic boundaries.
- You will learn to properly initialise a simulation.
- You will learn how to solve the equations of motion in discrete time steps.

2.2 Program structure

Whether the simulation applies to a system of molecules on the nanometer scale or to a system of granular particles on the meter scale, the general structure of a particle-

based program is usually the same:[1] there is a time loop in which forces on the particles are evaluated at a certain time, positions and velocities are updated to the next time step, and boundary conditions (walls or periodic boundaries) are applied. With the new velocities and positions, new forces are evaluated, etcetera.

The forces depend on relative distances and sometimes relative velocities between neighbouring pairs, triplets, etc. of particles. A large number of distances and velocities need to be evaluated if the number of particles is large, making the force evaluation step usually the most time-consuming part of the program. To reduce the computational cost it is advantageous to efficiently identify nearby particles by using neighbourlists or cell-linked-lists.

Of course the particle positions and velocities need to be initialised before the time loop, and saved when the required amount of simulation time T^{run} has been reached.

A flowchart outlining the structure of a general particle-based simulation program is given in Fig. 2.1. In the following sections we will focus on the essential features of each block. The preferred method of initialisation depends on the range of interaction between the particles and the system size and boundary conditions, and will therefore be discussed *after* discussing forces and boundary conditions.

2.3 Forces on particles

There are many different types of forces that may act upon a particle. Generally, we make a distinction between energy-conserving forces and dissipative forces, depending on whether energy is *apparently* conserved or not.

We can also make a distinction between **external forces** and **multi-body forces**. The distinction depends on what we define as our system of particles and what we designate as surroundings. An external force is a force that acts on a certain partice *irrespective* of the positions of the other particles in the system. Forces which do depend on the positions of the other particles are multi-body forces.

In the next subsections we will give examples and details of these various types of forces.

Conservative forces

Definition of conservative force

Conservative forces are most commonly encountered in the interactions between very small particles, such as Van der Waals interactions between molecules. At larger scales we can also find examples of conservative forces, such as Newton's force of gravity and

[1] In this course we focus on time-driven simulation programs for particles that interact through forces and in which particles have finite collision times. Time progresses with regular time steps. There exists another class of dynamical simulation programs for hard particles with instantaneous collisions. In these so-called event-driven simulations time progresses with irregular intervals from one collision event to the next.

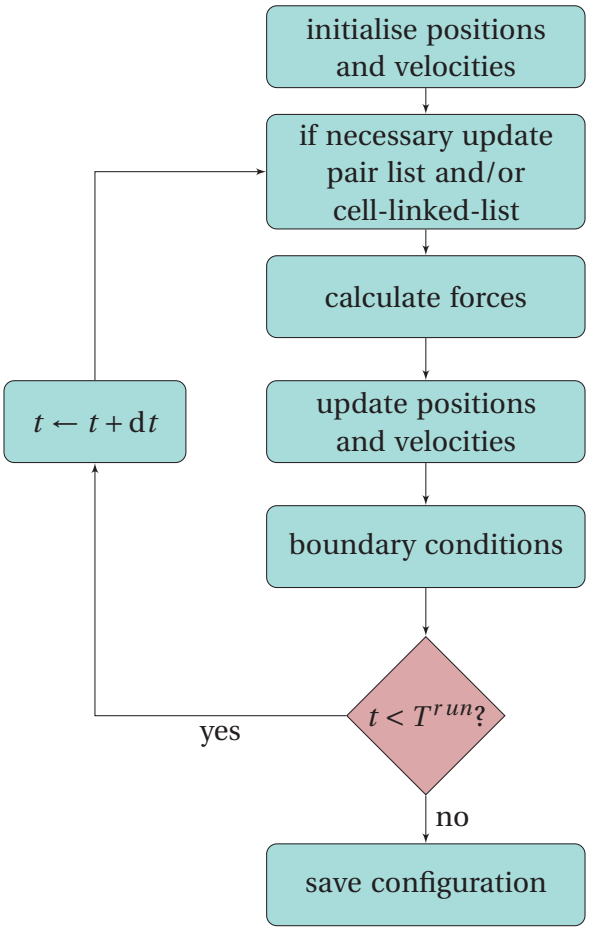

Figure 2.1: Flowchart of a general particle-based time-driven simulation program. If we are dealing with a continued run, the first block 'initialisation' means loading a previously saved configuration.

Coulomb's force between two charged particles. Often forces at large scales are idealised to conservative forces, such as the *elastic* forces associated with the deformation of a solid material (where plastic deformation or failure of the material is neglected).

Conservative forces can be derived from a potential energy Φ. This potential energy depends on the set of positions of the particles, but crucially *not* on their velocities: $\Phi = \Phi(\mathbf{r}_1, \ldots, \mathbf{r}_N)$. The force on an individual particle i is given by

$$\mathbf{F}_i = -\boldsymbol{\nabla}_i \Phi(\mathbf{r}_1, \ldots, \mathbf{r}_N), \tag{2.1}$$

where $\boldsymbol{\nabla}_i = \left(\partial/\partial x_i, \partial/\partial y_i, \partial/\partial z_i\right)$ is the gradient operator with respect to the position of particle i. To get a better feeling for what this means, imagine that we move a particle i taking various small excursions from its current position, while keeping the positions of all other particles fixed. Each small excursion leads to a different change in the total potential energy; Eq. (2.1) is telling us that the force on the particle points in the direction of steepest descent of the total potential energy.

Forces that can be derived from a potential energy like this are called conservative because they conserve total energy. This can easily be proven from Newton's equations

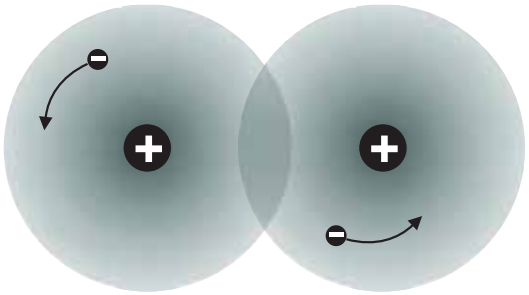

Figure 2.2: Pictorial representation of the interaction between two neutral spherical atoms. The nuclei (+) are much heavier than the electrons (-). In the Born-Oppenheimer approximation, the nuclei move in effective (electronically averaged) potentials. Nuclear translation, rotations and vibrations can therefore be treated by using classical mechanics.

of motion:

$$\begin{aligned} m_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} &= -\nabla_i \Phi \\ \sum_{i=1}^{N} m_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} \cdot \mathbf{v}_i &= -\sum_{i=1}^{N} \nabla_i \Phi \cdot \mathbf{v}_i \\ \frac{\mathrm{d}}{\mathrm{d}t}\left(\sum_i \frac{1}{2} m_i v_i^2\right) &= -\frac{\mathrm{d}}{\mathrm{d}t}\Phi \\ \Rightarrow \frac{\mathrm{d}}{\mathrm{d}t}\mathscr{H} &= 0 \qquad (2.2) \\ \mathscr{H} &= \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 + \Phi. \qquad (2.3) \end{aligned}$$

Eq. (2.2) shows that the total energy, expressed as the Hamiltonian $\mathscr{H}$, is a conserved quantity:

$$\mathscr{H}(t) = \mathscr{H}(0) = E. \qquad (2.4)$$

This does not mean that the potential energy and the kinetic energy are conserved quantities! There is a constant exchange of energy between the kinetic energy $K = \sum_i \frac{1}{2} m_i v_i^2$ and the potential energy Φ such that their sum is constant and equal to the total energy E.

Example: interatomic potential

The potential energy can often be derived from a fundamental knowledge of the relevant (usually chemical or physical) properties of the system. To give an explicit example, consider a group of N molecules interacting with each other. We will treat the simplest case, namely that of neutral spherical atoms. Noble gases such as argon and krypton are excellent examples. Additionally, we may treat nearly spherical molecules, such as methane, in a similar way.

First suppose we have just two atoms, fixed with their nuclei at positions $\mathbf{r}_1$ and $\mathbf{r}_2$, as in Fig. 2.2. We can write the total ground state energy of the two atoms as

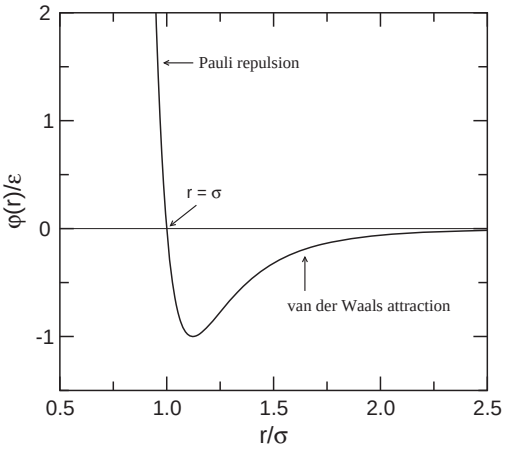

Figure 2.3: The total interatomic interaction between two neutral spherical atoms is well described by the Lennard-Jones formula, Eq. (2.7). At large distances the van der Waals attraction is dominant. At short distances the atoms repel each other because of the Pauli exclusion principle. The diameter of the atom may be defined as the distance σ where these two interactions exactly cancel out.

$$\epsilon_0(\mathbf{r}_1,\mathbf{r}_2) = \epsilon_0(\mathbf{r}_1) + \epsilon_0(\mathbf{r}_2) + \varphi(\mathbf{r}_1,\mathbf{r}_2). \tag{2.5}$$

Here $\epsilon_0(\mathbf{r}_1)$ is the ground state energy of atom 1 in the absence of atom 2, and similarly for $\epsilon_0(\mathbf{r}_2)$. So the term $\varphi(\mathbf{r}_1,\mathbf{r}_2)$ is the correction to the sum of two unperturbed ground state energies of the atoms. This term is also called the interatomic interaction or interatomic potential. Because of the rotational symmetry of the atoms, the interatomic potential only depends on the distance $r_{12} = |\mathbf{r}_1 - \mathbf{r}_2|$ between the two atoms, i.e.

$$\varphi(\mathbf{r}_1,\mathbf{r}_2) = \varphi(r_{12}). \tag{2.6}$$

It is also clear that because of its definition $\varphi(\infty) = 0$. At finite distances, the electrons in one atom will feel the electrons in the other atom. A classical picture would be the following: the charge distribution in an atom is not constant, but fluctuates in time around its average. Consequently, the atom has a fluctuating dipole moment which is zero on average. The instantaneous dipoles in the atoms, however, influence each other in a way which makes each dipole orient a little in the field of the other. This leads to the so-called van der Waals attraction between two neutral atoms. The van der Waals attraction becomes stronger as the atoms get closer to one another. At a certain point, however, the atoms will repel each other because of the Pauli exclusion principle. The total interatomic interaction as a function of distance is well described by the Lennard-Jones formula (see Fig. 2.3):

$$\varphi(r) = 4\epsilon\left\{\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right\}. \tag{2.7}$$

The parameter ϵ is the depth of the interaction well, and σ is the diameter of the atom. The values of ϵ and σ are characteristic for each atomic species. For example for argon $\epsilon/k_B = 117.7$ K and $\sigma = 0.3504$ nm, for krypton $\epsilon/k_B = 164.0$ K and $\sigma = 0.3827$ nm, and for methane $\epsilon/k_B = 148.9$ K and $\sigma = 0.3783$ nm. Here $k_B = 1.38065 \times 10^{-23}$ J/K is Boltzmann's constant. Note that at room temperature $T \approx 300$K, the magnitudes of ϵ are of the same order of magnitude as the thermal energy $k_B T$. This means that these atoms form a fluid (liquid or gas) at room temperature: the interatomic interactions are so weak that they allow the structural arrangement of the atoms to change dynamically under the influence of thermal fluctuations. This is hardly allowed in a solid.

Multiple particles

When dealing with more than two spherical atoms or molecules, it is often assumed that the total potential energy (due to particle interactions) may be approximated as a sum of pair interactions :

$$\Phi(\mathbf{r}_1,\ldots,\mathbf{r}_N) = \sum_{i=1}^{N-1}\sum_{j=i+1}^{N}\varphi(r_{ij}). \tag{2.8}$$

Note that the double sum is constructed such that each pair interaction is counted only once. In practice the pair-approximation is often a reasonable assumption.

We now ask ourselves: what is the force $\mathbf{F}_i$ on molecule i due to all these pair interactions? According to Eq. (2.1) we must take minus the gradient of the potential energy with respect to the position of molecule i:

$$\begin{aligned}\mathbf{F}_i &= -\boldsymbol{\nabla}_i\Phi \\ &= -\boldsymbol{\nabla}_i\sum_{j=1}^{N-1}\sum_{k=j+1}^{N}\varphi(r_{jk}) \\ &= -\sum_{j\neq i}\boldsymbol{\nabla}_i\varphi(r_{ij}).\end{aligned} \tag{2.9}$$

Going from the second to the third line we used the fact that the position $\mathbf{r}_i$ appears only in terms $\varphi(r_{jk})$ where either j or k is equal to i. There are exactly $N-1$ of these terms: the distances of all particles, except i itself, to particle i. We continue to evaluate the gradient $\boldsymbol{\nabla}_i$ of the intermolecular potential $\varphi(r_{ij})$ between a particular pair i and j. We will first consider one component, say the x-component:

$$\begin{aligned}\frac{\partial}{\partial x_i}\varphi(r_{ij}) &= \frac{\partial\varphi(r_{ij})}{\partial r_{ij}}\frac{\partial r_{ij}}{\partial x_i} \\ &= \varphi'(r_{ij})\frac{\partial}{\partial x_i}\sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2} \\ &= \varphi'(r_{ij})\frac{x_i-x_j}{\sqrt{(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2}} \\ &= \varphi'(r_{ij})\frac{x_i-x_j}{r_{ij}},\end{aligned} \tag{2.10}$$

where the prime indicates differentiation of the intermolecular potential with respect to its argument (the pair distance). Similar expressions hold for the y- and z-components. Combining everything, we can write the force $\mathbf{F}_i$ on molecule i as

$$\mathbf{F}_i = \sum_{j\neq i}\mathbf{F}_{ij} \equiv -\sum_{j\neq i}\varphi'(r_{ij})\frac{\mathbf{r}_{ij}}{r_{ij}}, \tag{2.11}$$

where $\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j$ and we have defined $\mathbf{F}_{ij}$ as the force *on* particle i due to the presence of particle j. We can interpret this expression as follows: for particles interacting

through radial pair potentials the *magnitude* of $\mathbf{F}_{ij}$ is given by minus the derivative of the the pair potential, and the *direction* is given by the unit vector $\mathbf{r}_{ij}/r_{ij}$ pointing from particle j to particle i.

When dealing with the force on particle j due to the presence of particle i the same pair distance between particles i and j will be encountered, leading to a contribution to the force on particle j that is exactly opposite the previous contribution to the force on particle i, i.e. $\mathbf{F}_{ji} = -\mathbf{F}_{ij}$. Therefore, in practice, each pair distance is evaluated once, and the forces on both i and j are updated using the above expression.

A code example: calculating forces between Lennard-Jones particles

A specific code example will make things clear. Suppose we have stored the positions of a collection of `N` Lennard-Jones particles as `x(i)`, `y(i)` and `z(i)`, with `i` the particle index. Suppose furthermore we have stored the squared particle diameter σ^2 in a variable `sigma2` and the energy ϵ in a variable `eps`. We can then calculate the total potential energy `Phi` and the force components `Fx(i)`, `Fy(i)` and `Fz(i)` on the particles with the following pseudo-code:

```
routine calculate_force
Phi = 0.0
Fx(:) = 0.0
Fy(:) = 0.0
Fz(:) = 0.0
do i=1,N-1
  do j = i+1,N
    xij = x(i)-x(j)
    yij = y(i)-y(j)
    zij = z(i)-z(j)
    rijsq = xij*xij+yij*yij+zij*zij
    sr6 = (sigma2/rijsq)^3
    sr12 = sr6*sr6
    Phi = Phi+4*eps*(sr12-sr6)
    Fr = 24*eps*(2*sr12-sr6)/rijsq
    Fx(i) = Fx(i)+Fr*xij
    Fy(i) = Fy(i)+Fr*yij
    Fz(i) = Fz(i)+Fr*zij
    Fx(j) = Fx(j)-Fr*xij
    Fy(j) = Fy(j)-Fr*yij
    Fz(j) = Fz(j)-Fr*zij
  enddo
enddo
end routine
```

Note that `Fr` is the force *divided* by the particle distance, i.e. $-\varphi'(r_{ij})/r_{ij}$. This way we only need to multiply by the distances along the respective cartesian components, $x_i - x_j$, $y_i - y_j$, $z_i - z_j$ to obtain the forces. We have also avoided calculating the distance r_{ij} directly (but rather used the squared distance r_{ij}^2), because taking a square root is computationally expensive. We could increase the computational speed even more by postponing the multiplication by constant factors such as 4ϵ until after the double loop.

External forces

External forces are forces on a particle that may depend on its position or velocity but do not depend on the positions or velocities of other particles within the system. Well-known examples include gravity force, and the force on magnetised or electrically charged particles due to an external magnetic or electric field.[2]

In many cases the external field can be written as a potential energy, in which case the results of the previous subsection apply again. For example, for a system of N particles moving in a constant gravity field, the potential energy due to gravity forces is

$$\Phi^e(\mathbf{r}_1, \ldots, \mathbf{r}_N) = \sum_{i=1}^{N} m_i g z_i, \tag{2.12}$$

where g is the gravitational acceleration, m_i the mass of particle i, and we have assumed that the gravity force is directed in the negative z-direction:

$$\mathbf{F}_i^e = -\boldsymbol{\nabla}_i \Phi^e = -m_i g \hat{\mathbf{e}}_z. \tag{2.13}$$

The gravity force is an external force because the total potential energy in Eq. (2.12) is a sum of terms which depend on individual particle positions, but crucially not on the relative particle positions.

Sometimes external forces cannot be written as a potential energy. For example, the Lorentz force on a charged particle i moving with a velocity $\mathbf{v}_i$ through an external electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$ is given by

$$\mathbf{F}_i^e = q_i \left(\mathbf{E} + \mathbf{v}_i \times \mathbf{B}\right), \tag{2.14}$$

where q_i is the charge of the particle. The force due to the magnetic field cannot be written as minus the gradient of a potential energy.[3] The Lorentz force is an external force because it does not depend on the positions or velocities of other particles within the system.

[2] Note that gravitational, magnetic and electric forces are also active between the particles within a system, but may be negligible compared to these forces due to interactions with matter outside the system, i.e. an *external* field.

[3] There are ways to include external electromagnetic forces as gradients and curls, using the so-called Lagrangian and magnetic vector potential, but this goes beyond the scope of this course.

Dissipative forces

Although the law of conservation of energy is generally valid, this is not always immediately apparent. When we are dealing with large objects (larger than molecules), we often ignore the detailed motion of particles inside and around these objects and instead take a more lumped (*coarse-grained*) view of an object.

Example: a swinging pendulum

For example, when a pendulum swings through the air, on a microscopic scale countless collisions take place between the atoms of the pendulum and the molecules in the air. Total energy is conserved in each and every collision. However, usually we do not wish to track the atoms in the pendulum and the molecules in the air, but rather describe the *effect* the air has on the motion of the pendulum which we treat as a solid body. On this level of description the air exerts a dissipative friction force on the pendulum, that tends to slow down its motion, and energy is apparently lost.

Example: a sphere moving through a liquid

A related example is the friction force experienced by a sphere moving through a stationary liquid such as water, which at low velocities scales linearly with the velocity:

$$\mathbf{F}_i^{fric} = -\zeta \mathbf{v}_i. \tag{2.15}$$

Here ζ (in units of kg/s) is the friction coefficient, which can be related to the water viscosity μ and the sphere radius a through $\zeta = 6\pi\mu a$, as we will see in Chapter 5. On a microscopic scale, the water molecules are continually colliding with the sphere and with each other. In rest, these collisions tend to cancel each other out.[4] However, when the sphere moves through the liquid, there will on average be more collisions at the front of the sphere than at the back. This explains why the friction force acts opposite the direction of motion. In the absence of external forces the all motion would eventually cease.[5]

Example: a block sliding down an inclined plane

A last example of a dissipative force due to collisions and interactions at smaller scales is the dynamic friction experienced by a dry object (say a block) sliding down an inclined plane. It is an empirical fact that this friction force scales linearly with the normal force N on the surface:

$$\mathbf{F}_i^{fric} = -\mu_f N \frac{\mathbf{v}_i}{|\mathbf{v}_i|}, \tag{2.16}$$

[4] More precisely, the average force is zero. The second moment of the force on the sphere is not zero; this leads to diffusion. We will discuss this also in Chapter 5.

[5] More precisely, the average velocity would tend to zero for the same reason as in the previous footnote.

where μ_f is the (dimensionless) coefficient of (Coulomb or kinetic) friction. On a microscopic scale the surfaces of the block and the inclined plane are not perfectly smooth, but rough. This rough surface may be viewed as a series of peaks and troughs. At small normal force, only the peaks of the surfaces physically touch each other, and it is relatively easy to transversally slide the two surfaces. At larger normal force, the surface material is deformed and a larger area is in direct physical contact. It is then much more difficult to slide the two surfaces past each other. To first order, the contact area increases linearly with normal force, which explains the equation given above. The term $-\mathbf{v}_i/|\mathbf{v}_i|$ indicates that this friction force also acts opposite the direction of motion. The coefficient of kinetic friction depends on the materials used; for example, ice on steel has a low coefficient of friction, while rubber on pavement has a high coefficient of friction.

Is energy really lost in all these examples?

Although it may appear that energy is lost in all these examples – without external forces all motion would eventually cease – it is important to realise that energy is not really lost, but rather converted into a more invisible form of energy, namely the kinetic energy of the fluid flow around the moving pendulum or sphere, and an increased intensity of the *random* motions of the molecules or atoms that constitute the gas, liquid or solid material. On a macroscopic scale, we say that some kinetic energy of the object has converted into kinetic energy of the fluid flow field and some kinetic energy has dissipated into heat, which which we measure as a slight increase in temperature of the air, water, pendulum, sphere or block.

2.4 Neighbourlists and cell-linked-lists

The evaluation of the forces is usually the most time-consuming part of a simulation code because this involves calculating the interactions between neighbouring pairs, triplets, etc. of particles. Here we will focus on pair interactions only, but the same techniques will apply to three-body forces.

A simple double loop, with i running from 1 to $N-1$ and j running from $i+1$ to N as we have done in the previous section to calculate the forces in a system of Lennard-Jones particles, requires the evaluation of $\frac{1}{2}N(N-1)$ pair distances. For a simulation containing 10.000 particles, this means that the computer needs to calculate about 50.000.000 distances each time step. This is very inefficient.

In many cases particles have a certain interaction range beyond which the interactions are zero, or so small that they may be neglected. Let us call this range the cut-off range r_{cut}. For particles interacting through the Lennard-Jones potential, see Fig. 2.3, we routinely choose $r_{cut} = 2.5\sigma$. In a system containing a large amount of particles, an overwhelming majority of particles is located at a distance larger than r_{cut} from any given particle. Moreover, in one time step the particle positions do not change much,

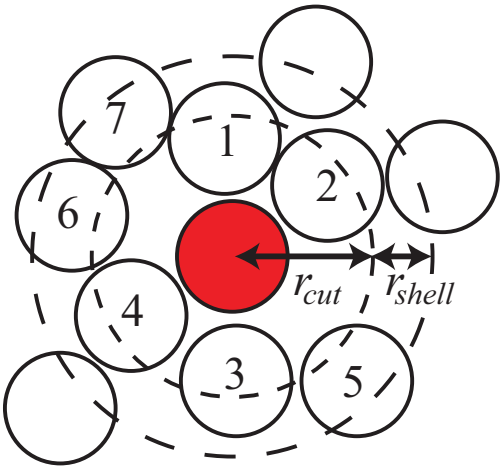

Figure 2.4: If the interactions between particles can be neglected beyond a cut-off range r_{cut}, it is advantageous to use a neighbourlist. This neighbourlist contains, for each particle (in this example indicated by the red particle), the indices of all neighbouring particles within the the cut-off range (here particles 1 to 4) *and* within an additional spherical shell of thickness r_{shell} (in this example particles 5 to 7). The additional spherical shell allows us to re-use the same neighbourlist for several time steps.

so a particle will be surrounded by the same set of closest neighbours for a considerable amount of time.

Neighbourlist

It now becomes apparent that it is computationally advantageous to create a list that contains the close neighbours of each particle [4]. This so-called neighbourlist can be re-used for several time steps (typically 10-50 time steps). The neighbourlist may be generated by evaluating all $\frac{1}{2}N(N-1)$ pair distances and storing, for each particle i, the indices of all particles that lie within a certain range $r_{list} = r_{cut} + r_{shell}$ of that particle, as shown in Fig. (2.4). The additional shell of thickness r_{shell} is necessary to already include neighbours in the neighbourlist that may enter the cut-off radius at a later time as they may move towards the central particle. The larger the shell thickness r_{shell}, the longer we can re-use the same neighbourlist. However, a thicker shell also implies that the total list range r_{list} is larger, and that a larger number of pair distances need to be evaluated each time step. It is therefore important to (empirically) find the shell thickness that results in the largest efficiency. For particles interacting through a Lennard-Jones potential, the optimal value of r_{shell} is usually in the range of 10% to 50% of the particle diameter σ.

No interactions between pairs of particles closer than r_{cut} should be missed. To ensure no pair interactions are missed, we monitor the particle displacement vectors $\mathbf{d}_i(t) = \mathbf{r}_i(t) - \mathbf{r}_i(t_{list})$ during the time a neighbourlist is re-used. The displacement vectors are initialised to zero when the neighbourlist is generated at time t_{list}, and updated during the general position update (which is discussed in section 2.7). From the set of displacement vectors we identify the largest displacement *length*: $d_{max} = \max\{|\mathbf{d}_1|, \ldots, |\mathbf{d}_N|\}$. The neighbourlist must be updated if:

$$d_{max} > r_{shell}/2 \qquad \text{(neighbourlist update necessary)} \tag{2.17}$$

The factor $\frac{1}{2}$ arises because two particles may be moving toward each other (think about this). Variations of this theme are possible. For example we could monitor the

largest and second largest displacement length and decide to update the neighbourlist if their sum is larger than the list radius. Because in practice the second largest displacement is not much smaller than the largest displacement, this does not really further improve the computational efficiency.

Cell-linked-list

If the number of particles is very large, say 10^5 or more, generating the neighbourlist by evaluating all pair distances is still computationally very demanding. In that case the efficiency of a simulation program is greatly improved by first sorting the particles into cells by means of a so-called cell-linked-list. Usually the cells have a length of r_{list}, or slightly more. In that case, to create the neighbourlist for a particular particle, it is only necessary to evaluate distances to particles within the same cell and particles in directly neighbouring cells (including diagonals). This means that in two dimensions only the particles in 8 neighbouring cells need to be checked, and in three dimensions 26 neighbouring cells (actually half these numbers, but more on that later).

The total number of cells in a system depends on the system size. If the system dimensions are `Lx` x `Ly` (in 2d) or `Lx` x `Ly` x `Lz` (in 3d), then the number of cells in each direction is given by

```
Mx = int(Lx/rlist)
My = int(Ly/rlist)
Mz = int(Lz/rlist)
```

where '`int`' is the truncated integer of its argument. The latter ensures that the cell size is always at least equal to r_{list}.

A larger system, containing a larger number of cells, will gain relatively more from the use of a cell-linked-list because a smaller fraction of the total number of particles needs to be processed when determining the distance to a certain particle. In practice, use of a cell-linked-list already becomes advantageous when the number of cells is at least 4 in each direction.

So how does a cell-linked-list look like? A cell-linked-list consists of two arrays: a `head` array containing the index of the first particle in a cell, and another `list` array containing, for each particle index, the index of the next particle within the same cell. An example is given in Fig. 2.5. Here we focus on a particular cell number 19. The first particle in this cell is `head`(19)=13. The next particle in cell number 19 can be found by looking up the value of `list`(13). The answer is 9. This continues, `list`(9)=7, `list`(7)=6, `list`(6)=4, until we hit `list`(4)=0, where the value zero indicates that particle 4 was the last particle in that particular cell. For an empty cell, for example cell 2 in Fig. 2.5, the value of `head` is already 0.

To generate these two arrays is surprisingly simple. If the x-coordinate of particle i is given by `x(i)`, which lies between 0 and `Lx`, and similarly for `y(i)` and `z(i)`, then in pseudocode:

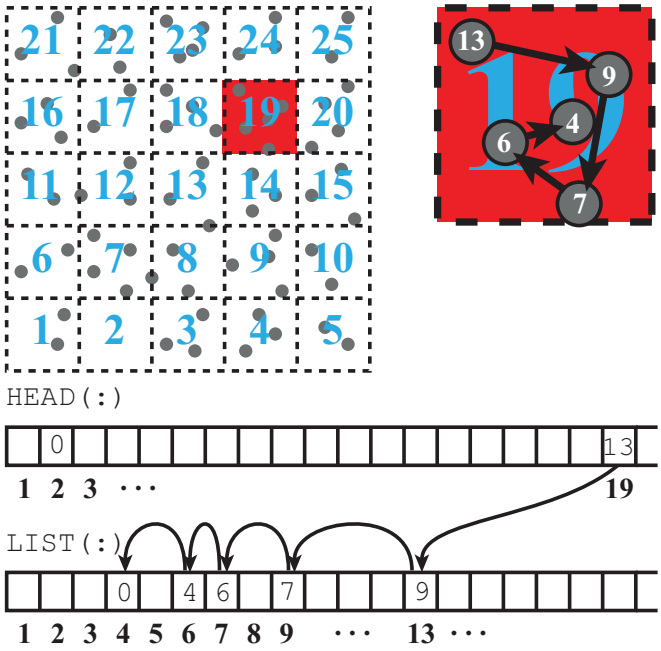

Figure 2.5: To generate a cell-linked-list, the particles are sorted into cells of size $\geq r_{list}$. When generating the neighbourlist of a certain particle, only pair distances with other particles in the same cell and with particles in directly neighbouring cells need to be evaluated. The actual cell-linked-list consists of two arrays: `head` and `list`. The contents of `head(icell)` is the index of the first particle in cell number `icell`. The contents of `list(i)` is the index of the next particle residing in the same cell as particle i.

```
routine generate_cell_linked_list
head(:) = 0
list(:) = 0
do i=1,N
  ix = int(x(i)*mxLx)
  iy = int(y(i)*myLy)
  iz = int(z(i)*mzLz)
  icell = 1 + ix + iy*Mx + iz*Mx*My
  list(i) = head(icell)
  head(icell) = i
enddo
end routine
```

Here `mxLx` = `Mx/Lx` is the inverse of the cell size in the x-direction, and similarly for the other directions. By using the above expression for the cell number, we have arranged the cells as indicated in Fig. 2.5. Another option is to make `head` a three-dimensional array, where `head(ix,iy,iz)` stores the first particle in cell `(ix,iy,iz)`.

It was already hinted that only half the neighbouring cells need to be checked. This is the case because the complete neighbourlist has to contain each unique particle pair only once. In two dimensions, the 4 cells to the top, the top-right, the right, and bottom-right need to be checked (think about why this choice is not unique). For example, when generating the neighbourlist of a particle in the red cell 19 in Fig. 2.5, only distances to particles in cells 19, 24, 25, 20 and 15 need to be checked. Because the cells do not move during the simulation, the indices of these 4 (in 2d) or 13 (in 3d) neighbouring cells can be calculated once and stored in an array, say `ngbcell`, in the

initialisation stage of a simulation. This will allow for a fast identification of neighbouring cells during the simulation. Calculating the neighbouring cells goes as follows in 2d:

```
routine initialise_ngbcell
ngbcell(:,:) = 0
do ix=0,Mx-1
  do iy=0,My-1
     icell = cell(ix,iy)
     ngbcell(icell,1) = cell(ix,iy+1)
     ngbcell(icell,2) = cell(ix+1,iy+1)
     ngbcell(icell,3) = cell(ix+1,iy)
     ngbcell(icell,4) = cell(ix+1,iy-1)
  enddo
enddo
end routine
```

The generalisation to 13 neighbouring cells in 3d is left as an exercise to the reader. The function `cell(ix,iy)` gives a unique index to each cell. When our system is periodic (as we will discuss later), there is a particularly easy way of generating the cell index. On the other hand, if the system is bounded, we must ensure that no cells beyond the system boundaries will appear as neighbouring cells. The following pseudo-code accomplishes this:

```
function cell(ix,iy)     !for bounded systems
  icell = 1 + ix + iy*Mx
  if (ix<0 .or. ix>Mx-1 .or. iy<0 .or. iy>My-1) icell = 0
  return icell
end function
```

Building a neighbourlist using a cell-linked-list

Suppose we have created a cell-linked-list. The next step is to actually generate the neighbourlist. When evaluating the pair distances of a certain particle we only need to check distances with particles that follow *after* that particle in the cell-linked-list of that cell (think about why!), and all particles in the 4 or 13 neighbouring cells. The following pseudo-code builds a neighbourlist `nbl(i,k)`, where i is the particle index, k the k'th neighbour of i, `nbl(i,k)` the index of this neighbour, and the *number* of neighbours of i is stored in `nbl(i,0)`. As discussed before, the neighbourlist should be updated as soon as the largest displacement of a particle in the system is more than half the shell thickness.

```
routine build_neighbourlist
nbl(:,:) = 0
do icell = 1,Mx*My
```

```
  i = head(icell)
  do while (i.ne.0)
    num_ngb_i = 0
    j = list(i)
    ################## all particles beyond i in current cell
    do while (j.ne.0)
      rij2 = (x(i)-x(j))^2 + (y(i)-y(j))^2
      if (rij2<rlist2) then
        num_ngb_i = num_ngb_i+1
        nbl(i,num_ngb_i) = j
      endif
      j = list(j)
    enddo
    ################## all particles in 4 neighbouring cells
    ################## (13 neighbouring cells in 3d)
    do i_ngbcell = 1,4
      jcell = ngbcell(i,i_ngbcell)
      do while (jcell.ne.0)
        j = head(jcell)
        do while (j.ne.0)
          rij2 = (x(i)-x(j))^2 + (y(i)-y(j))^2
          if (rij2<rlist2) then
            num_ngb_i = num_ngb_i+1
            nbl(i,num_ngb_i) = j
          endif
          j = list(j)
        enddo
      enddo
    enddo
    nbl(i,0) = num_ngb_i
    i = list(i)
  enddo
enddo
end routine
```

Here `rlist2` is the squared list range r_{list}^2. The above routine loops over each cell and then, for each particle i in that cell, generates a neighbourlist of i. This is accomplished first, by finding neighbouring particles in the rest of the cell-linked-list in which i is residing, and second by finding neighbouring particles in the neighbouring cells.

Calculating forces using the neighbourlist

Having obtained the neighbourlist, we can evaluate the pair interactions in the system for several time steps (this is why we went through all the trouble of generating

a neighbourlist after all!). For example for particles interacting through the Lennard-Jones interaction potential, as in subsection 2.3, we have in 2d and pseudo-code:

```
routine calculate_force_with_nbl
Phi = 0.0
Fx(:) = 0.0
Fy(:) = 0.0
do i = 1,N
  do k = 1,nbl(i,0)
    j = nbl(i,k)
    xij = x(i)-x(j)
    yij = y(i)-y(j)
    rijsq = xij*xij+yij*yij
    sr6 = (sigma2/rijsq)^3
    sr12 = sr6*sr6
    Phi = Phi+4*eps*(sr12-sr6)
    Fr = 24*eps*(2*sr12-sr6)/rijsq
    Fx(i) = Fx(i)+Fr*xij
    Fy(i) = Fy(i)+Fr*yij
    Fx(j) = Fx(j)-Fr*xij
    Fy(j) = Fy(j)-Fr*yij
  enddo
enddo
end routine
```

Note that in practice the efficiency of a simulation can be increased some more by already evaluating the pair interactions during the building of the neighbourlist (when the partice coordinates are available anyway). For sake of simplicity we have not done that here.

2.5 Boundary conditions

If the number and type of particles in a system have been specified, what happens inside a system is determined by the conditions that apply at the boundaries of the system. A simple example is the pressure of an amount of gas in a closed container, which depends on the volume enclosed by its impermeable walls. A more complicated example is the spatial evolution of the flowfield of a fluid forced to flow through a pipe, which depends on the imposed flow velocity at the inlet of the pipe (or alternatively the elevated pressure) and the amount of slip with the wall.

Conversely, if we are interested in the bulk properties of a system, the boundaries should be as far removed from the region of interest as possible, because the presence of walls is often felt over a length scale much larger than the typical particle-particle

interaction range. For example, in a Lennard-Jones liquid the radial distribution function (which we will encounter in detail in section 4.5) is typically still influenced by walls as far as 10 particle distances away from the walls. Therefore, in a 3d simulation containing N Lennard-Jones particles, only $(N^{1/3}-20)^3$ particles find themselves in a bulk-like region. This means that there is no bulk-like region in a box containing $N = 10^4$ particles, and that at least 10^6 particles are necessary to have at least 50% of the particles find themselves in a bulk-like region. In such a case it is helpfull to avoid walls altogether and use so-called periodic boundary conditions.

In the next subsections we will discuss the use of solid walls, flow boundary conditions, and periodic boundary conditions in particle-based simulations.

Solid walls

An obvious property of a solid wall is that no particles can penetrate through the wall. This could be accomplished through a potential, leading to an external force on each particle, or by direct modification of the velocity of a particle.

Wall-potentials

In microscopic (Molecular Dynamics) simulations it is customary to generate a wall through a wall-potential, although the same approach can be applied at any scale. When the detailed interactions with the walls are important for the problem, for example when dealing with wall-adsorption, a realistic but quite expensive option is to include a large number of wall particles, and freeze their positions, or connect them to each other or to lattice positions through spring forces. Note that this implies that the wall particles should also be included in the neighbourlists of the free particles.

For many problems, such an approach is too detailed and it suffices to mimick a wall through a potential than depends on the perpendicular distance between particle and wall. In detail, an additional energy term

$$\Phi^w(\mathbf{r}_1,\ldots,\mathbf{r}_N) = \sum_{i=1}^{N} \varphi^w(d_{iw}) \tag{2.18}$$

is added for each wall. Here d_{iw} is the closest distance between particle i and the wall. For a planar wall with unit normal $\hat{\mathbf{n}}$ (pointing inwards towards the system), $d_{iw} = (\mathbf{r}_i - \mathbf{r}_w)\cdot\hat{\mathbf{n}}$, where $\mathbf{r}_w$ is *any* point on the wall. For example, for a wall at $x = 0$, with the wall-normal in the positive x-direction, we have simply $d_{iw} = x_i$. For a wall at $x = L_x$, with the wall-normal in the negative x-direction, we have $d_{iw} = L_x - x_i$.

The wall potential $\varphi^w(d_{iw})$ should decay smoothly to zero at a finite range, and be sufficiently divergent for small distances to prevent the particle from ever reaching $d_{iw} = 0$. An often-employed example is the repulsive part of a Lennard-Jones-like potential:

$$\varphi^w(d) = 4\epsilon\left[\left(\frac{\sigma}{d}\right)^{2n} - \left(\frac{\sigma}{d}\right)^{n} + \frac{1}{4}\right] \qquad (\text{for } d < 2^{1/n}\sigma) \tag{2.19}$$

Check for yourself that this potential is purely repulsive, and smoothly decays to zero at $d = 2^{1/n}\sigma$. The parameter n controls the stiffness of the wall, where $n = 6$ corresponds to the stiffness of the Lennard-Jones interaction.

Because the walls act as an external force on each particle (i.e. independent of the positions of the other particles), the calculation of this type of wall force is relatively cheap and is best handled by a separate routine which loops once over each particle. For example, if we apply the following routine particles will be confined between two Lennard-Jones-like walls at $x = 0$ and $x = L$:

```
routine calculate_wall_force_and_energy
Phiwall = 0.0
do i=1,N
  xi = x(i)
  if (xi<rcutwall) then
    sx6 = (sigma/xi)^6
    sx12 = sx6*sx6
    Phiwall = Phiwall+4*eps*(sx12-sx6)
    Fx(i) = Fx(i)+24*eps*(2*sx12-sx6)/xi
  else if (xi>L-rcutwall) then
    sx6 = (sigma/(L-xi))^6
    sx12 = sx6*sx6
    Phiwall = Phiwall+4*eps*(sx12-sx6)
    Fx(i) = Fx(i)-24*eps*(2*sx12-sx6)/(L-xi)
  endif
enddo
end routine
```

Here `rcutwall`= $2^{1/6}\sigma$ is the cut-off distance for the wall interaction, and `Phiwall` stores the total wall potential energy.

The above wall potential leads to wall-induced forces on the particles which are oriented perpendicularly to the wall. In many cases this is sufficient, for example when one is interested in equilibrium properties such as the pressure of the system. However, it is important to realise that a planar wall potential cannot exert any force parallel to the wall. This means that flow of material parallel to the wall is uninhibited, corresponding to perfect slip boundary conditions at the continuum scale.

In some cases, especially when dealing with flow, it is important that a wall can sustain a certain amount of shear stress, which means that the particles moving parallel to the wall are slowed down by the wall. At the continuum scale this corresponds to no-slip (or partial-slip) boundary conditions. This can be achieved by including a large number of explicit wall particles and freezing their positions, or connecting them to each other or to lattice positions through spring forces. As mentioned before, this is computationally more expensive. A good alternative is to directly alter the velocities of the moving particles when they cross the boundary, as we will discuss next.

Solid walls through direct modification of particle velocities

Instead of explicitly simulating the entire collision process between a particle and a wall, where the wall-potential slowly grows and decays as the particle approaches and leaves the wall region, it is often sufficient to simply take into account the *effect* of a wall collision on the particle velocity.

The best known example is **specular reflection**. This kind of wall collision we know from our experience when a billiard ball bounces (without spin) against the edge of a billiard table. At the time of impact the velocity of the particle $\mathbf{v}$ is decomposed in a component $\mathbf{v}_\perp$ perpendicular to the wall and the remaining part $\mathbf{v}_{||} = \mathbf{v} - \mathbf{v}_\perp$ parallel to the wall. The effect of the collision is to invert the normal component, i.e. the new velocity is given by $\mathbf{v}' = \mathbf{v}_{||} - \mathbf{v}_\perp$. Because the parallel component $\mathbf{v}_{||}$ of the particle velocity is conserved, this situation again corresponds to perfect slip conditions at the continuum scale. The following pseudo-code gives an example where (non-rotating) spheres of diameter σ collide against walls at $x = 0$ and $x = L$ (i.e. they collide when the particle x-coordinate is equal to $\sigma/2$ or $L - \sigma/2$, respectively). For simplicity we assume that a particle `i` has moved on a straight path with velocity (`vx(i)`,`vy(i)`, `vz(i)`) during the last time step of length `dt`, so that we can simply mirror the x-position `x(i)` in $x = \sigma/2$ or $x = L - \sigma/2$:

```
routine specular_reflection
do i=1,N
  if (x(i)<0.5*sigma) then
    x(i) = sigma-x(i)
    vx(i) = -vx(i)
  else if (x(i)>L-0.5*sigma) then
    x(i) = 2*L-sigma-x(i)
    vx(i) = -vx(i)
  endif
enddo
end routine
```

In some simulations **bounce-back** conditions are applied. In this case the full velocity vector of the particle is inverted at the time of impact: $\mathbf{v}' = -\mathbf{v}$. It is easy to see that *on average* this leads to a zero velocity at the wall, i.e. no-slip conditions at the continuum scale. In pseudo-code, using the same example of spheres of diameter σ with bounce-back walls at $x = 0$ and $x = L$:

```
routine bounce_back
do i=1,N
  if (x(i)<0.5*sigma .or. x(i)>L-0.5*sigma) then
    if (x(i)<0.5*sigma) then
      dtremain = (x(i)-0.5*sigma)/vx(i)
    else
      dtremain = (x(i)-L+0.5*sigma)/vx(i)
```

```
    endif
    x(i) = x(i)-2.0*dtremain*vx(i)
    y(i) = y(i)-2.0*dtremain*vy(i)
    z(i) = z(i)-2.0*dtremain*vz(i)
    vx(i) = -vx(i)
    vy(i) = -vy(i)
    vz(i) = -vz(i)
  endif
enddo
end routine
```

Here `dtremain` is the remainder of the time step `dt`, after the time of collision, with which the particle needs to retrace its path. Although the bounce-back condition is rather strange when considering a single particle (imagine a particle exactly retracing its steps after colliding with a wall), it is acceptable when considering a large collection of particles interacting with the wall and with each other, because this will naturally lead to a statistical distribution in the impact velocities and angles.

In microscopic and mesoscopic simulations the statistical distribution of particle velocities inside the fluid is known: it is the Maxwell-Boltzmann distribution [45] shown in Figure 2.6:

$$P(v_x, v_y, v_z) = \left(\frac{m}{2\pi k_B T}\right)^{3/2} \mathrm{e}^{-\frac{m\left(v_x^2+v_y^2+v_z^2\right)}{2k_B T}}, \tag{2.20}$$

where m is the mass of the particle and k_B is Boltzmann's constant.[6] This distribution shows that a higher temperature T is associated with larger random velocity fluctuations. The idea behind **diffusive** or **thermal** wall conditions is the following: when we are dealing with small particles, we should also consider the morphology of the wall at microscopic scales. At small scales a wall is never perfectly smooth but rough and will contain numerous defects. This leads to a random scattering of the small particle back into the system. Also, the atoms that form the wall are not sitting still but performing thermal motions of themselves, dictated by the wall temperature and the atomic mass according to the Maxwell-Boltzmann equation above. When we consider an ensemble of many collisions between microscopic particles and such a wall, the ensemble of particles will equilibrate its kinetic energy with that of the wall. So the particles will attain the temperature of the wall. This may be modeled by drawing the new particle velocities from an appropriate thermal distribution. The particles are released into the fluid with this new velocity.

So what is the appropriate thermal distribution? The velocity distribution of the incoming particles (those that cross the wall boundary in a certain time interval) is not exactly the same as the Maxwell-Boltzmann distribution. They have a bias in their velocity component normal to the wall. This is easy to understand: only particles that

[6] Here we assume there is no background flow. If there is a background flow, **v** should be interpreted as the velocity *relative* to the background flow velocity, i.e. (v_x, v_y, v_z) are the velocity *fluctuations*.

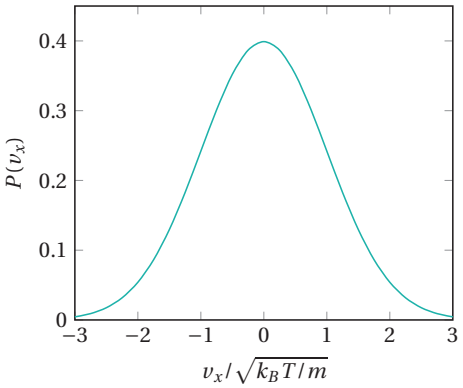

Figure 2.6: The probability of encountering a particle with velocity v_x along the x-direction in a thermal system is given by the Maxwell-Boltzmann distribution. This is a Gaussian distribution with zero average and standard deviation equal to $\sqrt{k_BT/m}$. The same applies to the other cartesian components v_y and v_z.

actually move towards the wall will collide with the wall. More precisely, particles with a larger velocity component $v_\perp$ toward the wall can originate from a larger distance $(v_\perp \Delta t)$ from the wall and still collide with it within a certain small time interval Δt. The probability for a normal component of the velocity distribution is therefore biased by an additional factor proportional to the normal component of the velocity. The components parallel to the wall are unaffected. In conclusion, the outgoing velocity components should be drawn from the following distributions (see Appendix C for details on how to generate such random numbers):

$$P(v_\perp) = \frac{m}{k_BT} v_\perp e^{-\frac{mv_\perp^2}{2k_BT}} \tag{2.21}$$

$$P(v_{||1}) = \sqrt{\frac{m}{2\pi k_BT}} e^{-\frac{mv_{||1}^2}{2k_BT}} \tag{2.22}$$

$$P(v_{||2}) = \sqrt{\frac{m}{2\pi k_BT}} e^{-\frac{mv_{||2}^2}{2k_BT}}. \tag{2.23}$$

Another advantage of using these boundary conditions is that the walls simultaneously act as a thermostat, draining or providing heat as needed to keep the temperature near the walls constant.

Inflow and outflow boundaries

If we study the properties of a system under flow, for example the convection and diffusion of a collection of colloidal particles in a liquid flowing through a pipe, we need to impose inflow and outflow boundary conditions.[7] There are many possible forms of inflow boundary conditions, depending on whether a certain pressure, a certain mass flux of particles and/or a precise velocity profile needs to be maintained at the entrance. Here we will describe the simple case of an imposed velocity profile.

[7] An often employed alternative to explicit inflow and outflow boundaries, especially useful for steady-state simulations with periodic boundaries (see next subsection) is to let the flow develop itself by exerting a constant external force $\mathbf{F}^{ext}$ on each fluid particle. This is equivalent to the effect of applying a pressure gradient $-\nabla P$ equal to $\rho^{\#}\mathbf{F}^{ext}$, where $\rho^{\#}$ is the number density of fluid particles.

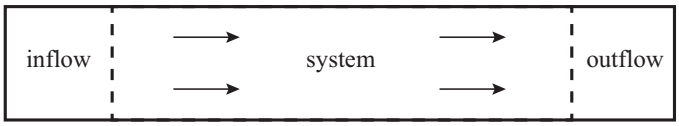

Figure 2.7: In simulations of flowing particles, new particles can be inserted in the inflow region and, after streaming through the system, removed again in the outflow region.

Inflow boundaries

Inflow boundary conditions are usually constructed as a region in space which acts as an entrance for new particles into the system of interest, see Figure 2.7. We will refer to this as the 'inflow region' from here on. Particles can be added with a certain mass rate Q_m (in kg/s) into the inflow region, and each particle is initialised with a certain flow velocity which depends on the desired inlet flow profile $\mathbf{v}^{inlet}(\mathbf{r})$.

The main problem usually is how to place the particles. For dilute and not too dense fluids we can achieve this by placing the particles randomly in the inflow region and continuously checking for (too large) overlap with the particles that have already been placed. For dense fluids this is often not possible and we need to resort to adding prepacked layers of particles, for example with a crystalline or nearly-crystalline structure, to achieve a high enough packing. To avoid sudden shocks in the forces on particles that are inside the system of interest, the interaction range or strength (e.g. σ or ϵ in the Lennard-Jones potential) may be gradually grown from a small initial value to the final value while the particle travels from its initial position in the inflow region to the entrance of the system of interest. We will discuss random insertion, crystalline packing and slow growth of interaction range and strength in more detail in section 2.6 on initialisation.

After a new particle i has been placed at $\mathbf{r}_i$, choosing its velocity is relatively straightforward. Depending on the location in the inflow region, the particle should get a velocity equal to the velocity of the desired flow profile at the location of the particle: $\mathbf{v}_i = \mathbf{v}^{inlet}(\mathbf{r}_i)$. For molecular and mesoscopic (i.e. thermal) systems appropriate thermal fluctuations $\delta\mathbf{v}_i$ should be added to this local average:

$$\mathbf{v}_i = \mathbf{v}^{inlet}(\mathbf{r}_i) + \delta\mathbf{v}_i, \tag{2.24}$$

where all three components of $\delta\mathbf{v}_i$ are drawn from a Maxwell-Boltzmann distibution:

$$P(\delta v_{i,x}) = \sqrt{\frac{m}{2\pi k_B T}} \mathrm{e}^{-\frac{m\delta v_{i,x}^2}{2k_B T}} \tag{2.25}$$

$$P(\delta v_{i,y}) = \sqrt{\frac{m}{2\pi k_B T}} \mathrm{e}^{-\frac{m\delta v_{i,y}^2}{2k_B T}} \tag{2.26}$$

$$P(\delta v_{i,z}) = \sqrt{\frac{m}{2\pi k_B T}} \mathrm{e}^{-\frac{m\delta v_{i,z}^2}{2k_B T}} \tag{2.27}$$

Remember that temperature is a measure for the local velocity *fluctuations*, so without adding these fluctuations the inflow of fluid will effectively be at a temperature of zero!

Outflow boundaries

Outflow boundary conditions are usually simpler than inflow boundary conditions. Often the only task is to remove particles from the system as soon as they cross the boundary. In some cases, especially in dense systems or in systems with long range interactions, removing a particle may lead to a sudden shock on the forces on the particles that are still inside the system of interest. If this is the case, we may choose to gradually decrease the interaction strength from the moment a particle moves into the outflow region (figure 2.7). When the interaction strength is small enough the particle can be removed entirely.

Complication: the number of particles is not constant!

There is a complication that must be taken into account when using inflow and outflow boundary conditions: not all particles are always inside the system plus inflow and outflow regions. Therefore the number of particles inside the system plus in- and outflow regions is not necessarily constant. If the largest possible number of particles inside the system can be estimated, this is not a very large problem. We can still define all our position and velocity arrays as before, but now add an extra boolean array (perhaps called `inside`) which indicates for each particle whether it is inside the system.[8]

When generating the cell-linked-list and/or neighbourlist, we should only process particles that are inside the system (`inside(i)=.true.`). Conversely, when new particles have been introduced to the system in the inlet region, we should add these particles to the appropriate cell-linked-lists and/or neighbourlists.

Example: flow of a gas between two parallel plates

A specific example will make things clear. Suppose, we study the flow of a dilute gas of Lennard-Jones particles of mass m, flowing in the positive x-direction between two parallel plates at $y = 0$ and $y = L_y$. Let us assume that the two plates are solid walls with no-slip boundary conditions with the gas, and that we feed the slit with a homogenous gas with a flat flow profile ($\langle v_x \rangle (y) = v_0 = \text{const}$). As the gas flows through the system, it will decelerate near the walls, and because of molecular collisions the slowing down effect will grow inwards.[9] If the desired initial number density is $\rho_0^\#$ (in two dimensions: the number of particles per unit area), the mass rate with which we must feed new particles is $Q_m = m\rho_0^\# v_0 L_y$. This means that on average we should each time step Δt add $n_{insert} = Q_m \Delta t / m = \rho_0^\# L_y v_0 \Delta t$ particles in the inlet region.[10] Because the gas is dilute we may place the particles at random, and there is no need to slowly

[8] Using a boolean array is generally more efficient than the other used practice where a particle is placed at some coordinates outside the system and the (real number) coordinates are used to check whether a particle is inside or outside the system.

[9] In other words: the boundary layer thickness will grow in time due to the viscosity of the fluid.

[10] Of course we can only insert an integer number of particles. We assume here that Q_m is so large that $n_{insert} \gg 1$ and simply taking the nearest integer of n_{insert} is sufficiently accurate. In the other extreme, if n_{insert} is smaller than 1, we can draw a uniform random number between 0 and 1 and insert

increase the interaction strength in the inflow region and decrease it in the outflow region. To prevent particles from escaping in the negative x-direction, we can insert a simple slip-boundary wall at the left side of the inflow region. A pseudo-code for all these boundaries is as follows.

```
routine inflow_outflow_boundary
########### inflow boundary
nindex = 0
i = 0
do while (nindex < ninsert)    !search for ninsert particles which
  i = i+1                      !are currently outside the system
  if (i>N) stop                !stop when we run out of particles
  if (inside(i)=.false.)
    placeparticle_inflow(i)    !place particle in the inflow region
    addparticle_nbl(i)         !add particle to neighbourlists
    vx(i) = v_0 + gauss(kT/m)  !gauss(x) gives a gaussian random number
    vy(i) = gauss(kT/m)        !with zero mean and variance x
    inside(i)=.true.
    nindex = nindex+1
  endif
enddo
########### bounce-back at y=0 & y=Ly, reflection at x=0, outflow at x=Lx
do i = 1,N
  if (inside(i)) then
    if (y(i)<0 .or. y(i)>Ly) then
      if (y(i)<0) then
        dtremain = y(i)/vy(i)
      else
        dtremain = (y(i)-Ly)/vy(i)
      endif
      x(i) = x(i)-2.0*dtremain*vx(i)
      y(i) = y(i)-2.0*dtremain*vy(i)
      z(i) = z(i)-2.0*dtremain*vz(i)
      vx(i) = -vx(i)
      vy(i) = -vy(i)
      vz(i) = -vz(i)
    endif
    if (x(i)<0) then
      x(i)  = -x(i)  !don't let particles escape
      vx(i) = -vx(i)
    endif
```

one particle if the random number is less than n_{insert}. In between these two extremes we could choose the actual number of particles to insert from a so-called Poisson distribution with an average of n_{insert}.

```
    if (x(i)>Lx) then
      inside(i) = .false.
      removeparticle_nbl(i)
    endif
  endif
enddo
end routine
```

In the first part relating to the inflow boundary, as long as the number of particles to be inserted has not been reached, we keep looking for the index of a particle outside of the system (`inside(i) = .false.`). When such a particle is found, new coordinates inside the inflow region are chosen, the neighbourlists are updated, a velocity is drawn from a Maxwell-Boltzmann distribution, and `inside(i)` is made `.true.`. If there are insufficient particles outside of the system (`i>N`) to accommodate the new particles in the inflow region, the program is stopped. For most modern programming languages this is not necessary, because they allow for a re-allocation of the arrays to a new (increased) length N.

In the second part, we recognize the specular and bounce-back reflection rules. When the x-coordinate of particle i becomes larger than the box size L_x, it is moved outside of the system by setting `inside(i)=.false.` and removing it from all neighbourlists. The more difficult operations are handled by routines `placeparticle_inflow`, `addparticle_nbl` and `removeparticle_nbl`.

The routine `placeparticle_inflow` places a particle inside the inflow region, which here is a rectangular region between $x = 0$ and $x = L_{inlet}$ and $y = 0$ and $y = L_y$. The new particle should not overlap with other particles. The check for overlap can be made efficient by again making use of the cell-linked-list, only searching for overlap with particles in nearby cells. Too keep this example short and simple, here we will simply loop over all particles.

```
routine placeparticle_inflow(i)
overlap = .true.
do while (overlap)
  xi = Linlet * ran()    !ran() gives a uniform random number on [0,1)
  yi = 0.5*sigma + (Ly-sigma) * ran()  !no overlap with solid y-walls
  overlap = .false.
  do j = 1,N   !note: using cell-linked-list would be more efficient
    if (inside(j)) then
      xij = xi-rx(j)
      yij = yi-ry(j)
      rijsq = xij*xij + yij*yij
      if (rijsq < rcut*rcut) overlap = .true.
    endif
  enddo
enddo
```

```
rx(i) = xi
ry(i) = yi
end routine
```

Note that the vertical position of the new particle is between $\sigma/2$ and $L_y - \sigma/2$, not between 0 and L_y, because the centre of a particle cannot approach the solid walls at $y = 0$ and $y = L_y$ closer than one particle radius $\sigma/2$.

A word of caution is at place here. The above routine will work for sufficiently dilute systems. In more dense systems the routine could get stuck looping endlessly trying to find a vacant position for a new particle. In that case, as mentioned before, we should not insert particles randomly but for instance stack them in well-arranged layers.

The routine `addparticle_nbl` adds the new particle to the neighbourlists. We could add the new particle to the neighbourlist of each particle placed within one list radius. However, because in the evaluation of the forces we treat each unique pair only once, we may as well insert existing particles in the neighbourlist of the new particle. Again, this could be done efficiently using a cell-linked-list, but for simplicity we will loop over all particles inside the system:

```
begin routine addparticle_nbl(i)
xi = rx(i)
yi = ry(i)
num_ngb_i = 0
do j = 1,N   !note: using cell-linked-list would be more efficient
  if (inside(j)) then
    xij = xi-rx(j)
    yij = yi-ry(j)
    rijsq = xij*xij + yij*yij
    if (rijsq < rlist*rlist) then !within list range of new particle
      num_ngb_i = num_ngb_i+1
      nbl(i,num_ngb_i) = j
    endif
  endif
enddo
nbl(i,0) = num_ngb_i
end routine
```

We loop over all particles and check first if the particle is inside the system and then whether it is within a distance `rlist` of the new particle i. You may worry that particle i itself will end up in the neighbourlist of i. This is not the case because at this point `inside(i)` is still `.false.`.

The routine `removeparticle_nbl` is the opposite of the previous routine: it removes a particle from all existing neighbourlists. When this particle is removed from a particular neighbourlist, in most cases it will create a hole in the neighbourlist. We can fill this hole with the last other particle of this neighbourlist. In pseudo-code:

```
begin routine removeparticle_nbl(i)
do j = 1,N            !note: inefficient simple loop for clarity
  if (inside(j)) then         !j=i is skipped: inside(i)=.false.
      loc_i = 0         !used to store location of i in nbl of j
      num_ngb_j = nbl(j,0)
      do k = 1,num_ngb_j
        if (nbl(j,k) = i) loc_i = k
      enddo
      if (loc_i .ne. 0) then
        nbl(j,loc_i) = nbl(j,num_ngb_j) !fill with last neighbour
        nbl(j,0) = num_ngb_j-1          !decrease no. neighbours
      endif
    endif
  endif
enddo
end routine
```

We loop over all particles and check first of the particle is inside the sytem and then whether one of the elements of the neighbourlist is equal to i. The location of i in the neighbourlist is stored in `loc_i` and the last neighbour in the neighbourlist is used to fill the hole created by the removal of i. Because at this point `inside(i)` is still `.true.`, the neighbourlist of i itself will not be checked.

We note that checking the neighbourlists of all particles is an expensive operation and should in practice be avoided, for example by identifying nearby particles using a cell-linked-list or, in case of an outflow boundary, by only checking the neighbourlists of particles within a distance r_{list} of the outflow boundary.

This concludes our treatment of inflow and outflow boundaries.

Periodic boundary conditions

Sometimes we are interested in the bulk properties of a system. In such a case, the boundaries should be as far removed from the region of interest as possible, because walls are known to structure a fluid over a length scale much larger than the typical particle-particle interaction range. Bulk behaviour may be approximated by the use of periodic boundary conditions.[11]

When we use periodic boundary conditions, exact copies of the central simulation box are generated in all directions, leading to an infinitely large grid of simulation boxes, see Figure 2.8. This has two consequences for the simulation:

[11] Bulk behaviour is only approximated by using periodic boundary conditions, because the number of degrees of freedom is still finite. One should therefore always test the effect of box size on the results. For example, near a phase transition and near a critical point, the characteristic size of density fluctuations in the fluid may grow beyond the size of a typical periodic simulation box. Also, hydrodynamic modes are limited to characteristic wavelengths commensurate with the simulation box size. This leads to subtle finite box-size effects which we will not discuss here.

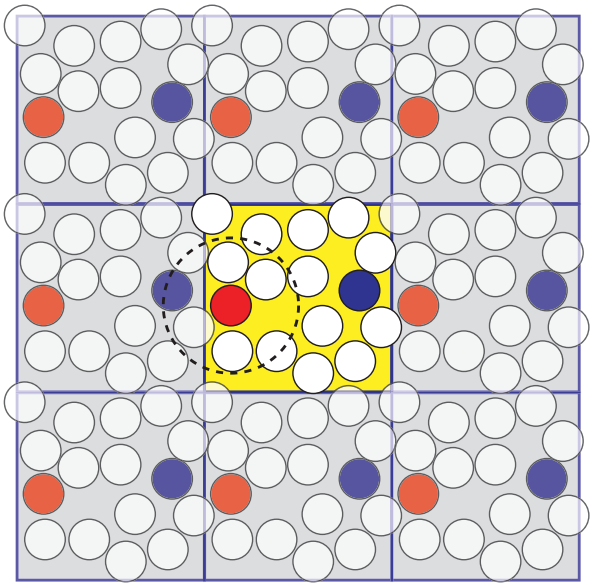

Figure 2.8: Explanation of periodic boundary conditions and the minimum image convention in a two-dimensional system. The contents of the central box (yellow) is copied in all directions. The red particle has a cut-off distance, indicated by the dashed circle, less than half the box size. The red particle in the central box therefore does not directly interact with the blue particle in the central box. Rather, it interacts with the closest image of the blue particle to its left side.

1. After the position update, when a particle in the central simulation box has left the box through the right boundary, it re-enters through the left boundary, and similarly in all directions.

2. During the calculation of forces, a particle not only interacts with neighbouring particles within the central box, but also with images of particles in neighbouring copies of the box.

To explain these two consequences in more detail, we suppose in the following that our box is two-dimensional and periodic in both directions, with the central box x-coordinates between 0 and L_x, and y-coordinates between 0 and L_y.

Applying periodic boundary conditions to the particle positions

As soon as the x-coordinate of a particle is larger than L_x, it has left the right boundary and should re-appear from the left boundary at an x-coordinate diminished by L_x. Conversely, when the x-coordinate is less than 0, we should add L_x to get the new x-coordinate. With a similar approach for the y-direction, we arrive at:

```
routine periodic_boundary_positions
do i=1,N
  if (x(i) >= Lx) x(i)=x(i)-Lx
  if (x(i) <  0 ) x(i)=x(i)+Lx
  if (y(i) >= Ly) y(i)=y(i)-Ly
  if (y(i) <  0 ) y(i)=y(i)+Ly
enddo
end routine
```

Depending on the computer hardware, the coding language, and the compiler, evaluating many if-statements can be relatively expensive. In that case a more elegant approach is to subtract an integer number of box lengths, where the number is determined by using the *nearest integer* (`nint`) evaluation.[12] If the central box is between $-L_x/2 \leq x < L_x/2$, applying a periodic boundary in the x-direction is rather simple: `x(i) = x(i)-Lx*nint(x(i)/Lx)` (find out yourself why this works). For a central box between $0 \leq x < L_x$, as we have assumed in the previous examples, we first subtract half the box length:

```
routine periodic_boundary_positions_with_nint
do i=1,N
  x(i) = x(i)-Lx*nint(x(i)/Lx-0.5)
  y(i) = y(i)-Ly*nint(y(i)/Ly-0.5)
enddo
end routine
```

Particle interactions with periodic boundaries: the minimum image convention

The use of periodic boundary conditions also implies that particles interact with images of particles in neighbouring copies of the central box. A particle may even interact with more than one copy of the same particle if the cut-off distance r_{cut} is larger than half the smallest box dimension (think about why?). This will both complicate the calculations and lead to severe finite box-size effects.

To avoid these complications, in practice we choose the smallest box size at least twice as large as he cut-off distance,

$$\min(L_x, L_y) \geq 2r_{cut}, \tag{2.28}$$

or when we use a neighbourlist twice the list distance:

$$\min(L_x, L_y) \geq 2r_{list}. \tag{2.29}$$

We can then use the so-called **minimum image convention**: in the calculation of the pair interaction between a pair of particles i and j we need only take into account *the closest image pair*.

To make this more explicit, let us consider an example of two particles i and j which happen to have the same y-coordinate. Suppose particle i is located at $x_i = 0.1L_x$ and a particle j at $x_j = 0.8L_x$, and that the cut-off distance is $r_{cut} = 0.4L_x$. The direct distance between particles i and j in the central box is $|x_i - x_j| = 0.7L_x$. Because this is larger than r_{cut} these particles do not interact directly inside the central box. However, the image of particle j *closest* to particle i is located at $x'_j = x_j - L_x = -0.2L_x$. The distance between particle i and this image of particle j is smaller than r_{cut}: $|x_i - x'_j| = 0.3L_x$. The force between i and j should be calculated based on this distance.

[12] For example, `nint(0.1)=0, nint(0.9)=1, nint(-0.1)=0, nint(-0.9)=-1`.

In the above example the two particles have the same y-coordinate. More generally, for particles interacting through a pair potential φ, the force on particle i due to interaction with particle j must be calculated as

$$\mathbf{F}_{ij} = -\varphi'\left(\left|\mathbf{r}_i - \mathbf{r}'_j\right|\right)\frac{\mathbf{r}_i - \mathbf{r}'_j}{\left|\mathbf{r}_i - \mathbf{r}'_j\right|}, \tag{2.30}$$

where $\mathbf{r}'_j$ is the image of j that is closest to i. To find this closest image we can again use the `nint` (nearest integer) function:

$$x_i - x'_j = x_{ij} - L_x \mathrm{nint}(x_{ij}/L_x), \tag{2.31}$$

where $x_{ij} = x_i - x_j$ is the displacement from particle j to particle i inside the central box. Similar expressions should be applied to the other (periodic) directions. In other words, *anywhere* in the simulation code where the vector from j to i is needed, the following lines:

```
xij = x(i)-x(j)
yij = y(i)-y(j)
```

should be replaced by:

```
xij = x(i)-x(j)
xij = xij-Lx*nint(xij/Lx)
yij = y(i)-y(j)
yij = yij-Ly*nint(yij/Ly)
```

It is important to emphasize that the minimum image convention should not only be applied when evaluating forces between particles, but also when building the neighbourlist, because particles at opposite sides of the box may actually be close together.

If the neighbourlist is generated through a cell-linked-list, we also need to adapt the function `cell(ix,iy)`. Remember that this function yields the index of a cell located at `(ix,iy)`. For a bounded system, a value zero was returned if `(ix,iy)` happened to fall outside the system boundaries. Now, when we search for neighbouring particles of a particle located near a periodic boundary, we should also check particles in cells located at the *opposite* side of the box. This may easily be achieved by using the modulo operator.[13] The adaptation then simply reads:

```
function cell(ix,iy)      !for periodic boundaries
  icell = 1 + mod(ix,Mx) + mod(iy,My)*Mx
  return icell
end function
```

With this adaptation, we can still use the same initialisation routine identifying neighbouring cells `initialise_ngbcell`. It is left as an exercise to the reader to find out, with the help of Figure 2.5, why this yields the correct neighbouring cell indices.

[13] The modulo $\mathrm{mod}(a,b)$ is defined as the (non-negative) remainder of a after division by b. For example $\mathrm{mod}(1,3) = 1$, $\mathrm{mod}(3,3) = 0$ and $\mathrm{mod}(-1,3) = 2$.

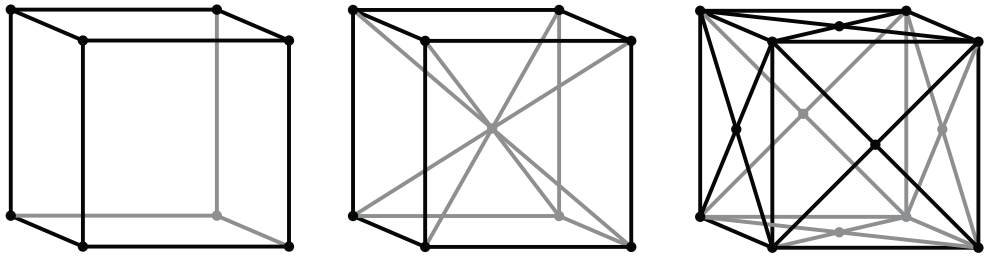

Figure 2.9: Primitive unit cells of cubic crystal arrangements.

2.6 Initialisation

The simulation has to start from a certain configuration of particles. So how do we choose the initial positions? This depends on the system at hand: we may place the particles in a lattice configuration or insert them randomly, possibly combined with slow growth of the particles. In all cases this should be followed by an equilibration simulation. In the following we will briefly discuss the applicability and advantages and disadvantages of these choices. Choosing the initial velocities of the particles is usually less complicated, and well known for microscopic (molecular) systems, as we will discuss at the end of this section.

Initialising on lattice positions

When dealing with an atomic or molecular crystal, we may place the particles in an appropriate lattice configuration that corresponds with the minimum (free) energy configuration of the particular material (under the desired thermodynamic conditions of temperature and/or pressure). Fig. 2.9 shows some common three-dimensional arrangements for spherical particles in a cubic periodic unit cell: the simple cubic, body-centered cubic (bcc), and face-centered cubic (fcc) configurations.

Even when we are not dealing with a crystal but with a liquid or dense gas, it is often advantageous to start with a crystal because this way potential problems with overlapping particles are easily avoided. When the total energy in the system is high enough (i.e. by choosing large enough initial velocities and/or by controlling the temperature through a thermostat), the equilibrium state will not be a crystalline solid but a liquid or gas. A certain amount of initial simulation time therefore needs to be allocated to ensure that the crystal will melt or evaporate. This is usually no problem for spherical particles, but may require a large amount of time for particles of non-trivial shape or particles with multiple attractive interaction sites, which may be associated with a high activation barrier for melting. In that case it may be better to randomly insert and slowly grow particles, as discussed in the following subsections.

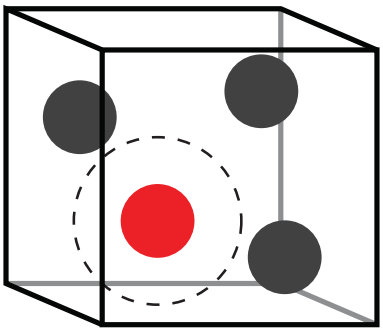

Figure 2.10: Random insertion of particles inside a simulation box. When attempting to place a new particle (red), we need to ensure that no previously placed particles are within a range r_{cr} (dashed line) of the new position. This only works well for relatively dilute systems.

Random insertion

We can be pragmatic and choose to simply insert the particles, one at a time, at random positions inside the simulation box. We already encountered this approach when we discussed a possible inflow boundary condition for a dilute gas of Lennard-Jones particles. If the center of the new particle ends up within a prescribed critical range r_{cr} of the centers of already placed particles, we try a new random position, see Fig. 2.10. An example pseudo-code is given below for the case of a two-dimensional periodic system of size `Lx` by `Ly`. The critical range can be chosen a little larger than the typical range of particle-particle interactions (the diameter for hard-sphere particles). As can be expected, this approach works well for relatively dilute particle configurations, but quickly fails when the particle density gets higher.

```
routine initialise_random_insertion
do i = 1,N
  overlap = .true.
  do while (overlap)
    xi = Lx * ran()   !ran() gives a uniform random number on [0,1)
    yi = Ly * ran()
    overlap = .false.
    do j = 1,i-1      !check overlap with already placed particles
      xij = xi-rx(j)
      yij = yi-ry(j)
      xij = xij-Lx*nint(xij/Lx) !minimum image convention
      yij = yij-Ly*nint(yij/Ly)
      rijsq = xij*xij + yij*yij
      if (rijsq < rcr*rcr) overlap = .true.
    enddo
  enddo
  rx(i) = xi
  ry(i) = yi
enddo
end routine
```

Growing particles or shrinking boxes

For very dense systems it is not always possible to place the particles by random insertion. Besides initialising on a lattice, another possibility is to initialise the system by deliberately starting with smaller particles or by placing the particles in a too large box. Subsequently, the size of the particles is increased in small steps or the box dimensions are shrunk in small steps, always with intermittent equilibration runs.

For example, for the case of a Lennard-Jones system, we can choose the initial particle size σ at a value of 50% of its final value. This allows us to place the particles by random insertion without too many rejections and without causing the subsequent equilibration run to immediately diverge. Actually, during the equilibration run the various energies (kinetic and potential) should be monitored until they reach a steady state. Then σ can be increased to 60% of its final value, and another equilibration run is executed. Etcetera, until the desired particle size is reached.

The other possibility is to choose the initial box dimensions too large, for example at 150% of the final box dimensions. This way there is also enough room for all particles to be placed by random insertion. After an equilibration run, the box dimensions are rescaled by a factor x close to unity (e.g. 0.9). When rescaling the box dimensions, of course the particle coordinates should be rescaled as well, $\mathbf{r}_i \rightarrow x\mathbf{r}_i$, and another equilibration run is executed. Etcetera, until the desired box dimensions are reached.

Both methods to generate a dense system are valid. Both methods are in a sense also the same: scaling of the particle size σ is equivalent to scaling of the box size L because L/σ is the important parameter.[14]

Choosing initial velocities

Before the time loop of a simulation program can commence, the particles should be initialised with a velocity. For thermal systems in equilibrium we should use the Maxwell-Boltzmann distribution, which we have by now encountered several times, but repeat here for completeness. For a particle of mass m_i the velocities are distributed according to:

$$P(v_{i,x}) = \sqrt{\frac{m}{2\pi k_B T}} \mathrm{e}^{-\frac{m_i v_{i,x}^2}{2k_B T}} \tag{2.32}$$

$$P(v_{i,y}) = \sqrt{\frac{m}{2\pi k_B T}} \mathrm{e}^{-\frac{m_i v_{i,y}^2}{2k_B T}} \tag{2.33}$$

$$P(v_{i,z}) = \sqrt{\frac{m}{2\pi k_B T}} \mathrm{e}^{-\frac{m_i v_{i,z}^2}{2k_B T}} \tag{2.34}$$

[14] Similarly, scaling of the interaction strength ϵ is equivalent to another equilibration method we have not discussed, namely a gradual temperature quench in which the system is cooled down in small steps from a too high temperature to the desired temperature. Here $\epsilon / k_B T$ is the important parameter.

It is often desirable to initialise the system with a total momentum of zero, meaning that the centre of mass of the system does not move. This may be achieved by subtracting the centre of mass velocity from the velocity of each particle. This requires two loops over the particles. The following pseudo-code gives an example for a two-dimensional system with particles of different mass m_i.

```
routine initialise_velocities
vcomx   = 0.
vcomy   = 0.
totmass = 0.
do i = 1,N
  vx(i)   = gauss(kT/m(i)) !gauss(x) gives a gaussian random number
  vy(i)   = gauss(kT/m(i)) !with zero mean and variance x
  vcomx   = vcomx + m(i)*vx(i)
  vcomy   = vcomy + m(i)*vy(i)
  totmass = totmass+m(i)
enddo
vcomx = vcomx/totmass
vcomy = vcomy/totmass
do i = 1,N
  vx(i) = vx(i)-vcomx
  vy(i) = vy(i)-vcomy
enddo
end routine
```

We note that in some simulation programs, after removing the centre-of-mass velocity, the velocities of the particles are rescaled to ensure that the kinetic energy of the system equals exactly $\frac{d}{2}Nk_BT$, where d is the number of dimensions. However, if we take the point of view that the total kinetic energy should consist of $\frac{1}{2}k_BT$ per degree of freedom (equipartition of energy for quadratic terms in the Hamiltonian [45]), such a rescaling is not necessary because the number of degrees of freedom is $d(N-1)$ instead of dN: when calculating the expected (ensemble average) kinetic energy we should take into account that conservation of the centre of mass momentum removes d degrees of freedom.[15]

[15] An extreme example would be a system consisting of two particles ($N = 2$) of mass m in one dimension ($d = 1$). Choosing both v_1 and v_2 from a Maxwell-Boltzmann distribution will generally lead to a non-zero centre of mass velocity ($v_{com} = (v_1 + v_2)/2 \neq 0$). Subtracting v_{com} from both velocities leads to an (ensemble averaged) kinetic energy of $\langle E_{kin}\rangle = \frac{m}{2}\left\langle (v_1 - v_{com})^2 + (v_2 - v_{com})^2 \right\rangle = \frac{m}{2}\left\langle \frac{1}{2}v_1^2 - v_1 v_2 + \frac{1}{2}v_2^2 \right\rangle = \frac{m}{2}\frac{k_BT}{m} = \frac{1}{2}k_BT$. Indeed the number of degrees of freedom in this case is equal to $d(N-1) = 1(2-1) = 1$.

2.7 Updating particle positions and velocities: numerical integration of the equations of motion

Now almost all elements of our basic particle-based simulation program are in place. We have initialised the particle positions and velocities, and built routines to calculate forces on the particles and deal with various types of boundaries. To actually determine the dynamics of the system, i.e. the time-dependence of the positions and velocities of the particles, we need to solve Newton's equations of motion:[16]

$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i, \tag{2.35}$$

$$m_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{F}_i, \tag{2.36}$$

where m_i is the mass of particle i and $\mathbf{F}_i$ is the sum of all forces on particle i, both due to interactions with other particles and due to external forces. The equations of motion are solved numerically. Rather than evaluating time derivatives based on infinitesimal time steps, we will evaluate time derivatives using a small but finite time step Δt.

Maximum time step is limited by the fastest oscillation

The time step Δt should be at least small enough to properly resolve the smallest oscillation times T_{min} present in the system, i.e. Δt should be at least, say, 20 times smaller than T_{min}. The smallest oscillation time may be estimated by evaluating the largest *curvature* of the potential energy (effectively the largest spring constant k_{max}) between a pair of particles. Using the analogy of a harmonic oscillator, the minimum oscillation time is never smaller than:

$$T_{min} = 2\pi\sqrt{\frac{m}{k_{max}}} = 2\pi\sqrt{m\left(\left.\frac{\mathrm{d}^2\varphi}{\mathrm{d}r^2}\right|_{max}\right)^{-1}} \tag{2.37}$$

where m is the (smallest) mass of a particle. This equation shows that heavier particles or particles interacting with softer potentials allow for larger integration time steps. What is meant by "practically reacheable curvature" depends on both the pair interaction and the typical relative velocities (i.e. the temperature for microscopic thermal systems) of the particles. For thermal systems we may estimate the curvature at a typical maximal potential energy of the order of $5k_BT$.

The curvature considerations lead to an upper limit of the time step. However, the accuracy of the numerical integration of the equations of motion depends on the type of time discretisation.

[16] We will encounter a different kind of equation of motion when we treat Brownian Dynamics in Chapter 5.

First order Euler method

The simplest approach is called the **first order Euler method**:

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t)\,\Delta t, \tag{2.38}$$

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t) + \frac{1}{m_i}\mathbf{F}_i(t)\,\Delta t \tag{2.39}$$

A disadvantage of this method is that it does not conserve the total energy (even when the forces are conservative), as will be discovered in the Practicum. Comparing Eq. (2.38) with the exact Taylor expansion around time t,

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \dot{\mathbf{r}}_i(t)\Delta t + \frac{1}{2}\ddot{\mathbf{r}}_i(t)\Delta t^2 + \frac{1}{6}\dddot{\mathbf{r}}_i(t)\Delta t^3 + \ldots, \tag{2.40}$$

shows that the truncation error of the first order Euler algorithm is quadratic in Δt. Generally, a method is called of order n if the truncation error scales as Δt^{n+1}.

Leap-frog method

If we increase the accuracy of the calculation by decreasing the time step, then at equal total simulation time the required number of time steps will increase, and hence the required calculation time. Therefore, it is advisable to choose an inherently more accurate algorithm to integrate the equations of motion.

One particularly popular algorithm is the so-called **leap-frog method** [4]. It may be derived as follows: performing a Taylor expansion *back* in time, we find

$$\mathbf{r}_i(t-\Delta t) = \mathbf{r}_i(t) - \dot{\mathbf{r}}_i(t)\Delta t + \frac{1}{2}\ddot{\mathbf{r}}_i(t)\Delta t^2 - \frac{1}{6}\dddot{\mathbf{r}}_i(t)\Delta t^3 + \ldots. \tag{2.41}$$

Adding Eq. (2.41) to Eq. (2.40), and rearranging, we find

$$\mathbf{r}_i(t+\Delta t) = 2\mathbf{r}_i(t) - \mathbf{r}_i(t-\Delta t) + \ddot{\mathbf{r}}_i(t)\Delta t^2 + \mathcal{O}\left(\Delta t^4\right), \tag{2.42}$$

Note that this is a third order algorithm with the favourable property that it is time-reversible: when at a certain time all Δt's are changed to $-\Delta t$'s, the particles will move back along their exact previous trajectories. This is not the case for the first order Euler algorithm and the core reason why that algorithm does not conserve total energy. To apply this algorithm in practice, we need to store the positions of the particles at a previous time step $\mathbf{r}_i(t-\Delta t)$; only forces ($\ddot{\mathbf{r}}_i(t) = \mathbf{F}_i(t)/m_i$) and no velocities are needed for the update of the particle positions. In many cases we would also like to have available the velocities, for example to calculate the kinetic energy. It is therefore convenient to first define a new variable $\mathbf{v}_i(t+\Delta t/2)$, such that

$$\mathbf{r}_i(t+\Delta t) - \mathbf{r}_i(t) \equiv \mathbf{v}_i(t+\Delta t/2)\Delta t. \tag{2.43}$$

Using this definition, we may rewrite the integration algorithm as:

$$\mathbf{v}_i(t+\Delta t/2) = \mathbf{v}_i(t-\Delta t/2) + \frac{\mathbf{F}_i(t)}{m_i}\Delta t, \tag{2.44}$$

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t+\Delta t/2)\Delta t. \tag{2.45}$$

It is important to stress that, despite the introduction of the new variable $\mathbf{v}_i$, numerically the integration of the position is still equal to that of Eq. (2.42), and therefore of third order. Instead of storing the positions at a previous time step, we now need to store the new variable, which is an *approximation* of the velocity at time $t+\Delta t/2$, and we therefore refer to loosely at "the" velocity. The above algorithm is called the leap-frog scheme because of the way in which the velocity is updated to $t+\Delta t/2$ using the force at time t, and subsequently the position is updated to $t+\Delta t$ using the just-obtained velocity at time $t+\Delta t/2$. Algorithmically, the position and velocity updates are very similar to that of the first order Euler algorithm. The difference is subtle (find it for yourself), but important!

The leap-frog algorithm leads to (approximate) velocities at half-time steps. The velocity at a whole time step, say time t, can be estimated as soon as $\mathbf{v}_i(t+\Delta t/2)$ is known, by taking following average:

$$\mathbf{v}_i(t) \approx \frac{1}{2}\left[\mathbf{v}_i(t-\Delta t/2) + \mathbf{v}_i(t+\Delta t/2)\right]. \tag{2.46}$$

The so-obtained velocities are accurate up to second order in Δt.

Velocity Verlet method

Sometimes it is more convenient to have the velocity at time t directly available, instead of having to calculate it through Eq. (2.46). In that case we can use the **velocity Verlet** method, which numerically is the same as the leap-frog algorithm [4]. Suppose we know the position and velocity at a certain time t, the velocity Verlet method then proceeds to the next time step as follows:

$$\mathbf{v}_i(t+\Delta t/2) = \mathbf{v}_i(t) + \frac{1}{2}\frac{\mathbf{F}_i(t)}{m_i}\Delta t, \tag{2.47}$$

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t+\Delta t/2)\Delta t, \tag{2.48}$$

(then evaluate force at $t+\Delta t$)

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t+\Delta t/2) + \frac{1}{2}\frac{\mathbf{F}_i(t+\Delta t)}{m_i}\Delta t. \tag{2.49}$$

Note that the force only needs to be evaluated once per time step, because $\mathbf{F}(t+\Delta t)$ in one time step is equal to $\mathbf{F}(t)$ in the next.

Also note that the higher order accuracy of both leap-frog and velocity Verlet methods are retained only as long as the force at time t only depends on the positions of the particles at time t, and not on the velocities at time t. For dissipative systems (where the force does depend on the velocity), the accuracy is generally lower, i.e. smaller time steps need to be made to achieve a sufficient level of accuracy.

2.8 Practicum: Debye crystal

In this practicum you will code your first full simulation program of a harmonic oscillator dumbbell (two particles) without friction. You will check various numerical schemes for energy conservation. Then you will extend the code to N coupled harmonic oscillators with periodic boundaries. This so-called Debye model is a model for a crystal. Finally, you will investigate the influence of non-harmonic interactions between the particles.

CHAPTER 3

DIMENSIONLESS NUMBERS AND SCALES

3.1 Chapter objectives

Through the course of this chapter, you will accomplish the following:

- You will learn about common physical phenomena that may be relevant in your simulations.
- You will learn about dimensionless numbers that characterise ratios of forces or transport properties.
- You will learn to distinguish between the three main types of particle-based simulations, namely microscopic, mesoscopic and macroscopic simulations.

3.2 Physical phenomena

> "Everything should be made as simple as possible… but not simpler."
> - Albert Einstein

When performing particle-based simulations, it is important to focus on what question you try to answer. As a simulator it is your responsibility to assess the relevance of various physical phenomena and make the right kind of approximations. As an extreme and obvious example, if you want to study circulating patterns in the path of granular particles in a fluidised bed, it does not make sense to include the velocity fluctuations of the atoms in the granular particles. Rather, it makes more sense to include the granular particles as single entities with a small number of degrees of freedom (position

and orientation) and certain collision properties. The above quote warns that, conversely, we should neither simplify the problem too much. In our example, we could for simplicity leave out the particle interactions and instead study the balance between gravity and hydrodynamic drag on a single particle. However, in such a case no circulating patterns will appear because the interactions (contact forces and hydrodynamic forces) are essential for disturbing particle from a simple one-dimensional path.

In the next subsections we will describe some common forces and transport phenomena. This list is by no means exhaustive, but will give you an impression of the different factors that may be taken into account in a simulation. Where appropriate we will give typical values for particles in water or air at room temperature and under standard atmospheric conditions.

Gravity and buoyant forces

All particles have a mass and therefore attract each other through gravitational forces. Unless we are dealing with the motion of stars or planets, it is usually sufficient to treat the gravity as an external force acting on each particle and ignore mutual gravitational attraction. On the surface of the Earth, a particle of mass m will feel a gravitational force

$$\mathbf{F}_g = -mg\hat{\mathbf{e}}_z \tag{3.1}$$

with $\hat{\mathbf{e}}_z$ pointing away from the Earth centre and $g = 9.81\ \mathrm{m/s^2}$.

Gravitation also causes a density and pressure gradient in fluids; under hydrostatic (equilibrium) conditions the pressure in a fluid changes with depth as $\nabla P = -\rho_f g\hat{\mathbf{e}}_z$, where ρ_f is the (local) fluid density. For example, $\rho_f = 1.0 \times 10^3\ \mathrm{kg/m^3}$ for water and $\rho_f = 1.2\ \mathrm{kg/m^3}$ for air under atmospheric conditions. For solid particles of volume V_p embedded in a fluid, this pressure gradient leads to a buoyant force

$$\mathbf{F}_b = -V_p \nabla P = V_p \rho_f g\hat{\mathbf{e}}_z. \tag{3.2}$$

So the total effect of gravity on the particle is determined by the difference between the particle density ρ_p and the fluid density ρ_f,[1] $\mathbf{F}_g + \mathbf{F}_b = -V_p\left(\rho_p - \rho_f\right) g\hat{\mathbf{e}}_z$.

Viscous and drag forces

In general, for a shear deformation rate of $\dot{\gamma}$ in a fluid, the viscous force exerted on an area A parallel to the flow direction is given by $\mu A\dot{\gamma}$, where μ is the dynamic viscosity (units Pa.s = kg/(m.s)). For water $\mu = 10^{-3}$ Pa.s, while for air under atmospheric conditions $\mu = 1.8 \times 10^{-5}$ Pa.s.

The viscosity is an important parameter determining the drag on a particle moving with velocity $\mathbf{V}$ through a stationary fluid. Because the fluid cannot penetrate the particle, the liquid bounces off the particle and needs to flow around the particle, leading

[1] Sometimes this *total* force is referred to as the buoyant force, so always check the definition.

to both a pressure distribution and a shear flow near the surface of the particle. At low particle velocities (how low will be made more exact later), the flow field in the fluid surrounding the particle scales linearly with the particle velocity V, and consequently the drag force will scale linearly with the particle velocity.[2]

We can make an order of magnitude estimate for the drag force on a single spherical particle of radius R moving at low velocity. The typical shear deformation rate is of the order of V/R and the (relevant) sphere area scales as R^2. We therefore expect the drag force $\mathbf{F}_d$ to scale as μRV. An exact calculation, performed by Stokes in 1851, shows that the drag on a sphere with no-slip boundaries is given by:

$$\mathbf{F}_d = -6\pi\mu R\mathbf{V} \qquad (V \text{ low}) \tag{3.3}$$

This is the famous *Stokes' law*. A full derivation is given in Appendix A.2.

Inertial forces

Continuing with the previous example, at higher particle velocities two things happen. First, the boundary layer (in which the fluid velocity changes from zero to V near the particle surface) decreases in thickness, leading to a faster-than-linear scaling of the typical shear deformation rate. Second, the inertia of the fluid which is suddenly accelerated in front of the particle becomes increasingly important. Indeed, if the particle velocity V becomes very high, the drag force on the particle will be dominated by the inertia of the fluid. Again we can make an order of magnitude estimation. The fluid at the frontal area A of the particle will be accelerated. Every second an amount of fluid of mass $\rho_f VA$ is accelerated to a velocity V. This leads to an inertial force of the order of $\rho_f V^2 A$. In general, the relation between drag force on a particle and its velocity is given by

$$\mathbf{F}_d = -C_d \frac{1}{2}\rho_f A|\mathbf{V}|\mathbf{V} \tag{3.4}$$

with C_d the so-called drag coefficient. For a smooth sphere $A = \pi R^2$ and $C_d \approx 0.44$ at high velocities, meaning that the drag force increases with the *square* of the particle velocity.

Surface tension

Molecules generally attract each other through cohesive forces (e.g. the Van der Waals r^{-6} part of the Lennard-Jones potential). In the bulk of a liquid, the molecules are pulled equally in all directions, leading to a net force of zero. However, when the liquid is in contact with a gas or another (immiscible) liquid, the molecules at the interfacial surface feel different forces toward the two sides of the surface; this imbalance in

[2] In this so-called Stokes flow regime the hydrodynamic equations, governing the motion of the fluid, are linear.

forces may be counteracted by curving the interface. Well-known examples include the spherical interface of a water droplet floating in air (in the absence of gravity) or a droplet of oil floating in water. In all cases, the interface has a "tension", i.e. tends to contract and minimize the surface area.

The surface tension γ is defined as the force along a line of unit length, where the force is parallel to the interfacial surface but perpendicular to the line. In terms of energy, an interface of area A between two phases is associated with an energy γA. Clearly, surface tension is not a property of a single fluid, but that of a fluid and another phase. For example, the surface tension between water and air is $\gamma = 7.2 \times 10^{-2}$ N/m.

One of the consequences of surface tension is that the pressure inside a droplet of a fluid is larger than the pressure outside the droplet. A balance between surface tension and pressure difference ΔP is achieved when

$$\Delta P = \frac{2\gamma}{R}, \tag{3.5}$$

where R is the radius of the droplet. This equation, known as the Young-Laplace equation, shows that the internal pressure increases when the droplets get smaller.

Diffusive transport of mass, momentum or energy

Many particle-based simulations are ultimately concerned with transport of mass, momentum or energy from one location to the other. This transport can occur through two main mechanisms, namely diffusive or convective. In the next subsection we treat convection, which is the process by which mass, momentum or energy is transported due to the mean motion of the fluid in which it is carried. In this subsection we treat diffusion, which is the process by which material is transported by the random thermal motion of the molecules within the fluid, even in the absence of any mean flow.

Mass diffusion

Small particles such as molecules and colloidal particles perform random motions in fluids, driven by thermal fluctuations. As a consequence, concentration differences in these particles tend to equalise. Defining c_i as the concentration of particle species i (in mol/m^3), the molar flux $\mathbf{J}_{c,i}$ (in mol/(m^{2}s)) is proportional to the gradient in particle concentration (*Fick's law*):

$$\mathbf{J}_{c,i} = -D_i \boldsymbol{\nabla} c_i, \tag{3.6}$$

where the constant of proportionality D_i is called the diffusion coefficient of species i (in units m^2/s). Combining the above equation with the conservation equation of mass, $\partial c_i/\partial t + \boldsymbol{\nabla} \cdot \mathbf{J}_{c,i} = 0$, yields the diffusion equation:

$$\frac{\partial c_i}{\partial t} = D_i \nabla^2 c_i. \tag{3.7}$$

Fick's law applies to the transport of many particles simultaneously, induced by concentration gradients. Similar arguments may be made for the time evolution of the probability distribution $P(\mathbf{r}, t; \mathbf{r}_0, t_0)$ to find a *given tracer particle* near location $\mathbf{r}$ at time t, if it was located at $\mathbf{r}_0$ at time t_0. This leads to an equation similar to Fick's law, $\partial P/\partial t = D_s \nabla^2 P$, but where D_s is the so-called self-diffusion coefficient.[3] It is easy to understand that the initial condition for this differential equation is $\lim_{t\to 0} P(\mathbf{r}, t; \mathbf{r}_0, t_0) = \delta(\mathbf{r} - \mathbf{r}_0)$,[4] indicating that the probability becomes more peaked around $\mathbf{r}_0$ for shorter intervals $t - t_0$. The solution of this equation is the so-called Green's-function for the (free) diffusion equation. In three dimensions:

$$P(\mathbf{r}, t; \mathbf{r}_0, t_0) = \frac{1}{(4\pi D_s(t - t_0))^{3/2}} \exp\left[-\frac{(\mathbf{r} - \mathbf{r}_0)^2}{4D_s(t - t_0)}\right] \tag{3.8}$$

We can use Green's function to study the ensemble dynamics of a diffusing particle. The ensemble average position of the particle at time t is

$$\langle \mathbf{r}(t) \rangle = \int P(\mathbf{r}, t; \mathbf{r}_0, t_0) \mathbf{r} \, \mathrm{d}^3 r = \mathbf{r}_0, \tag{3.9}$$

meaning that on average the particle remains where it was initially located at t_0. So what happens to the mean-square displacement of the particle? The mean-square displacement is a measure of the degree of fluctuations in the particle's position, given by the second moment of Green's function:

$$\left\langle (\mathbf{r}(t) - \mathbf{r}_0)^2 \right\rangle = \int P(\mathbf{r}, t; \mathbf{r}_0, t_0)(\mathbf{r} - \mathbf{r}_0)^2 \, \mathrm{d}^3 r = 6 D_s t. \tag{3.10}$$

In other words, the mean-square-displacement is proportional to time and the self-diffusion coefficient. The self-diffusion coefficient of water at room temperature is 2.3×10^{-9} m^2/s and that of oxygen in air is 1.8×10^{-5} m^2/s.

In summary, even in the absence of flow, small particles will move on a random path. Although the average vector displacement is zero, the typical root-mean-square distance from the original position grows as $\sqrt{t}$.

Thermal diffusion

We first treat thermal diffusion because the equations are very similar to those of mass diffusion. As the Maxwell-Boltzmann equations (2.20) show, the temperature T is a measure for the random velocity fluctuations of molecules and other small particles.

[3] At not too high particle densities, the diffusion coefficient of Fick's law (which applies to concentration) is equal to the self-diffusion coefficient. At high concentrations individual particles may become trapped while collective rearrangements still lead to an equalisation of the concentration. In that case the self-diffusion coefficient is lower than the (collective, Fick) diffusion coefficient.

[4] Here $\delta(\mathbf{r})$ is the three-dimensional Direc-delta function, which is zero everywhere except at the origin, and has the property $\int \mathrm{d}^3 r \delta(\mathbf{r}) = 1$.

Because these particles interact with each other, temperature differences between different regions in the system will be equalised. This equalisation is perceived as a heat flux $\mathbf{q}$ (in W/m^2), proportional to the gradient in temperature (*Fourier's law*):

$$\mathbf{q} = -k\boldsymbol{\nabla} T, \tag{3.11}$$

where the constant of proportionality k is called the thermal conductivity coefficient (in units W/(m.K)). The thermal conductivity coefficient of water is 0.60 W/(m.K) and that of air 0.025 W/(m.K).

By the divergence theorem, a small volume V of material will loose a net heat of $V\boldsymbol{\nabla}\cdot\mathbf{q}$ per second, and the temperature of the material in that volume will decrease (for positive $\boldsymbol{\nabla}\cdot\mathbf{q}$). This link between temperature decrease and heat loss (per unit volume) is provided by the enthalpy H: $\mathrm{d}H = \rho c_p \mathrm{d}T$, where ρ is the mass density (in kg/m^3) and c_p the specific heat capacity of the material (in J/(kg.K)). Assuming k is constant, we get:

$$\frac{\partial H}{\partial t} = \frac{\partial \rho c_p T}{\partial t} = -\boldsymbol{\nabla}\cdot\mathbf{q} = k\nabla^2 T \tag{3.12}$$

If we can also assume that the product of specific heat capacity and density is approximately constant, we find the so-called heat equation:

$$\frac{\partial T}{\partial t} = \alpha\nabla^2 T, \tag{3.13}$$

where $\alpha = k/(\rho c_p)$ is the thermal diffusivity (in units m^2/s). The thermal diffusivity of water is 1.4×10^{-7} m^2/s and that of air 1.8×10^{-5} m^2/s.

Momentum diffusion

We previously discussed viscous drag forces, and introduced the dynamic viscosity μ to characterise shear forces on an area (parallel to the flow direction) caused by shear flow. These shear forces, like mass diffusion and thermal conductivity, are determined by molecular processes.

Let us now analyse the situation in some more detail. Consider a fluid between two very large parallel plates which are some distance apart along the y-direction. The plates are so large that we may neglect end-effects, meaning that all physics is the same in the x- and z-directions. If the upper plate is moved with a velocity V along the x-direction, collisions with the fluid molecules immediately adjacent to this plate will accellerate the molecules in the x-direction too. Thermal motion and molecular collisions in the fluid will cause this effect to grow downwards, all the way to the bottom plate. As long as the fluid at a larger y-position moves with a higher x-velocity than the fluid below, the fluid below will be accelerated in the x-direction (and the fluid above decelerated). In other words, x-momentum will continually diffuse down through the fluid from the upper plate, i.e. in the negative y-direction. For simple fluids such as

water and gases, the flux of x-momentum τ_{yx} (in N/m^2 = Pa) passing every second through an imaginary plane *with normal along* y, i.e. y = const, is proportional to the local gradient in the velocity field (*Newton's law*):

$$\tau_{yx} = -\mu \frac{\partial v_x}{\partial y}, \tag{3.14}$$

where the constant of proportionality μ is the dynamic viscosity (in units Pa.s = kg/(m.s)). The minus sign occurs because for a positive $\partial v_x/\partial y$ the flow of x-momentum is in the negative y-direction.

Now a flux of x-momentum is nothing but a force per unit area acting in the x-direction (to convince yourself, just consider the units). The net shear force on a volume of fluid enclosed by imaginary planes at y and $y+\mathrm{d}y$ of an area A (in the xz-plane) is given by the momentum flux over the plane at y *minus* the momentum flux over the plane at $y+\mathrm{d}y$ (minus, because the inwards direction is pointing down at $y+\mathrm{d}y$). So we have: $F_x^{shear} = A\tau_{yx}(y) - A\tau_{yx}(y+\mathrm{d}y) \approx -A(\partial\tau_{yx}/\partial y)\mathrm{d}y$. Combining this with Newton's law, we find a net shear force density of $f_x^{shear} = \mu(\partial^2 v_x/\partial y^2)$. This net shear force can be used to accelerate the fluid:[5]

$$\rho \frac{\partial v_x}{\partial t} = \mu \frac{\partial^2 v_x}{\partial y^2} \tag{3.16}$$

which we may also write as $\partial v_x/\partial t = \nu(\partial^2 v_x/\partial t^2)$, with $\nu = \mu/\rho$ the kinematic viscosity. The kinematic viscosity of water is 1.0×10^{-6} m^2/s, that of air is 1.5×10^{-5} m^2/s.

Note that the kinematic viscosity ν in Eq. (3.16) has the same units as the mass diffusion coefficient D in Eq. (3.7) and thermal diffusivity α in Eq. (3.13). All these phenomena are diffusive, meaning that their extent will grow with time as $\sqrt{t}$. Also note that in a gas diffusive transport is mainly taking place because of the motion of the particles, not because of interactions between molecules. As a consequence, in a gas all three diffusion coefficients are of the same order of magnitude. Indeed, we have found that the self-diffusion of oxygen (very similar to the self-diffusion of nitrogen) is $D_s = 1.8\times10^{-5}$ m^2/s, the thermal diffusivity of air is $\alpha = 1.8\times10^{-5}$ m^2/s, and the kinematic viscosity of air is $\nu = 1.5\times10^{-5}$ m^2/s.

Convective transport of mass, momentum or energy

Convective transport is transport as a consequence of macroscopic motion of the fluid, i.e. *flow*. Suppose the fluid is flowing locally with a velocity $\mathbf{v}$. For a general conserved

[5] Note that we have derived an incomplete and non-general version of the Navier-Stokes equation here. For a general incompressible ($\boldsymbol{\nabla}\cdot\mathbf{v}=0$) flow of Newtonian fluids, the Navier-Stokes equation is:

$$\rho\left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{v}\right) = -\boldsymbol{\nabla}P + \mu\nabla^2\mathbf{v} + \rho\mathbf{g}, \tag{3.15}$$

where P is the pressure, and $\rho\mathbf{g}$ the force density due to gravitational forces (or any other external forces).

quantity Y (which can be moles, mass, momentum or thermal energy), we can define the local density X (molar concentration c, mass density ρ, momentum density $\rho\mathbf{v}$ or thermal energy density $\rho c_p T$). The amount of X that flows through a unit area perpendicular to $\mathbf{v}$ every second, i.e. the convective flux, is then equal to $\mathbf{J}_X^{conv} = X\mathbf{v}$. Explicitly, the convective concentration flux, convective mass flux, convective momentum flux and convective heat flux are given by

$$\mathbf{J}_{c,i}^{conv} = c_i\mathbf{v} \tag{3.17}$$
$$\mathbf{J}_{m,i}^{conv} = \rho\mathbf{v} \tag{3.18}$$
$$\mathbf{J}_{\rho\mathbf{v}}^{conv} = \rho\mathbf{v}\mathbf{v} \tag{3.19}$$
$$\mathbf{q}^{conv} = \rho c_p T\mathbf{v} \tag{3.20}$$

Here we have used the symbol $\mathbf{q}$ for heat flux. Note that almost all these fluxes may be represented by vectors with elements (J_x, J_y, J_z), i.e. they have a magnitude and a direction. There is one exception: the convective momentum flux is a tensorial quantity of second order, i.e. it may be represented by a matrix with elements $J_{xx} = \rho v_x^2, J_{xy} = \rho v_x v_y, J_{xz} = \rho v_x v_z$ on the first row, and similarly for the second and third row. In general, an element $J_{\alpha\beta} = \rho v_\alpha v_\beta$ represents the convective flux of α-momentum transported in the β-direction.[6]

Convective heat flux

If the fluid is moving past a solid surface, and the fluid has a temperature T_f different from the solid surface temperature T_s, a thermal boundary layer may develop. In this thermal boundary layer a temperature profile exists due to the energy exchange resulting from the temperature difference. The heat flux is in that case given by

$$q^{conv} = h(T_s - T_f), \tag{3.21}$$

where h is defined as the heat transfer coefficient (in W/(m^2.K)).

Convective mass flux

Similarly, if the fluid is moving past another phase (an immiscible fluid or a porous solid), and the fluid has a certain species concentration different from the concentration in the other phase, a mass-transfer boundary layer may develop. In this mass-transfer boundary layer a concentration profile exists due to the mass exchange resulting from the concentration difference. The concentration flux (mass transfer) is in that case given by

$$J_{c,i}^{conv} = K\Delta c_i, \tag{3.22}$$

where Δc_i is the concentration difference (mol/m^3) and K is defined as the mass transfer coefficient (in units of m/s).

[6] Since the convective momentum flux is a symmetric tensor, we directly find that the convective flux of α-momentum transported in the β-direction is equal to the convective flux of β-momentum in the α-direction.

Propagation of sound

Sound waves are longitudinal waves of momentum, whose distance from a momentum source grows linearly in time. In other words, sound waves are pressure waves which move with a certain velocity, the *speed of sound* c_s, through the system. The speed of sound is directly related to the compressibility of a material:

$$c_s^2 = \frac{\partial P}{\partial \rho} \tag{3.23}$$

The speed of sound in water is $c_s = 1.48 \times 10^3$ m/s, while the speed of sound in air is $c_s = 3.43 \times 10^2$ m/s.

Sound waves behave very different from the diffusive transport of momentum we encountered before,[7] for two reasons. First, sound waves cannot transport net momentum; the longitunal waves correspond to mere oscillations in local momentum. Second, the sound waves travel as $c_s t$ from the source of momentum, while viscous effects travel as $\sqrt{\nu t}$. This means that in most practical situations sound waves are felt long before the viscous effects are felt. Can you estimate at what distance from a momentum source in water the sound waves and viscous effects are felt simultaneously?

3.3 Dimensionless numbers

We mentioned before that it is your responsibility as a simulator to assess the relevance of various physical phenomena and make the right kind of approximations for your model. Making this assessment is greatly facilitated by estimating various dimensionless numbers. These dimensionless number are usually ratios of two types of forces or two types of transport properties. If a dimensionless number is much smaller or much larger than unity, one of the two physical phenomena may possibly be neglected.

Now, given the large number of physical phenomena, the number of dimensionless numbers is very large. To create some structure in this zoo of dimensionless numbers, we present in Table 3.1 the dimensionless numbers associated with ratios of common physical forces, and in Table 3.2 the dimensionless numbers associated with ratios of common transport properties.

In the following we give an (alphabetical) list of the most common dimensionless numbers encountered in systems that can be treated in particle-based simulations. The meaning of the symbols is the following:

- L is a characteristic size of a particle (e.g. the diameter of a sphere) [m].
- V is a characteristic velocity (of a particle or fluid, depending on context) [m/s].
- D is the self-diffusion coefficient of a particle or the molecular diffusion coefficient in the fluid, depending on context [m^2/s].

[7] Diffusive transport of momentum, i.e. viscous diffusion, is associated with transversal waves of momentum.

	inertia	viscosity	surface tension	gravity
inertia		1/Re	1/We	1/Fr
viscosity	Re		1/Ca	Ar, Gr
surface tension	We	Ca		Eo
gravity	Fr	1/Ar,1/Gr	1/Eo	

Table 3.1: Dimensionless ratios of physical forces (force mentioned in top row divided by force mentioned in left column).

	mass diffusion	momentum diffusion	thermal diffusion	mass convection	heat convection
mass diffusion		Sc	Le	Pe, Sh	
momentum diffusion	1/Sc		1/Pr		
thermal diffusion	1/Le	Pr		Pe_f	Nu
mass convection	1/Pe, 1/Sh		$1/Pe_f$		
heat convection			1/Nu		

Table 3.2: Dimensionless ratios of transport properties (transport property mentioned in top row divided by transport property mentioned in left column).

- ρ_p is the mass density of the particle [kg/m^3].
- ρ_f is the mass density of the fluid [kg/m^3].
- $\Delta\rho$ is the mass density difference between two fluids [kg/m^3].
- γ is the surface tension between two fluids [kg/s^2].
- μ is the dynamic viscosity of the fluid [kg/(m.s)].
- ν is the kinematic viscosity of the fluid [m^2/s].
- β is the thermal expansion coefficient of the fluid [1/K].
- k is the thermal conductivity coefficient of the fluid [W/(m.K)].
- α is the thermal diffusivity of the fluid [m^2/s].
- h is the heat transfer coefficient between fluid and solid (particle) [W/(m^2.K)].
- K is the mass transfer coefficient between the fluid and another phase [m/s].
- λ_{free} is the mean free path of a fluid molecule [m].
- g is gravitational acceleration (or acceleration due to a similar external force) [m/s^2].

Archimedes number Ar

The Archimedes number is the ratio of gravitational forces to viscous forces on a particle, taking into account the buoyancy:

$$\mathrm{Ar} = \frac{gL^3\rho_f(\rho_p - \rho_f)}{\mu^2}. \tag{3.24}$$

When $\mathrm{Ar} \gg 1$ natural convection dominates, i.e. less dense particles rise and denser particles sink in the fluid. When $\mathrm{Ar} \ll 1$ forced convection dominates, i.e. the particle tends to follow the flow of the fluid.

Capillary number Ca

The capillary number represents the relative effect of viscous forces versus surface tension forces acting across an interface between a liquid and a gas or between two immiscible liquids:

$$\mathrm{Ca} = \frac{\mu V}{\gamma}. \tag{3.25}$$

For low Ca (typically $\mathrm{Ca} < 10^{-5}$), flow in porous media is dominated by capillary forces.

Eötvös number Eo

The Eötvös number represents the relative effect of gravity forces (taking into account the buoyancy) to surface tension forces acting on a bubble or droplet moving in a fluid:

$$\mathrm{Eo} = \frac{\Delta\rho g L^2}{\gamma}. \tag{3.26}$$

When $\mathrm{Eo} < 1$, the surface tension dominates and the bubble or droplet may be approximated by a sphere. When $\mathrm{Eo} \gg 1$, the bubble or droplet is relatively unaffected by surface tension effects, generally leading to non-spherical shapes when they rise or sink through the fluid.

Froude number Fr

The Froude number is used in a number of ways. In general it represents the ratio of a characteristic velocity V to a gravitational wave velocity c,

$$\mathrm{Fr} = \frac{V}{c}. \tag{3.27}$$

The gravitational wave velocity depends on the application. For example, for gravity induced surface waves in shallow fluids (such as tidal waves) of uniform depth d we

have Fr = $V/\sqrt{gd}$. The Froude number may also be viewed as a ratio of a particle's inertia to gravitational forces.

In rotating equipment rotating with an angular velocity Ω, another (local) Froude number is defined, which signifies the relative importance of centrifugal force to gravitational force. For a fluid at distance R from the axis of rotation, we have Fr = $\Omega^2 R/g$.

When Fr < 1 the flow is subcritical, meaning no shock waves are generated. When Fr > 1 shock waves emerge, similar to the shock waves encountered in supersonic flow (see Mach number).

Grashof number Gr

The Grashof number is the ratio of buoyancy to viscous force acting on a particle or droplet due to a heat transfer, i.e. a difference between temperature T_p of the particle and the temperature T_f of the (bulk) fluid,

$$\mathrm{Gr} = \frac{g\beta(T_p - T_f)L^3}{\nu^2}. \tag{3.28}$$

The Grashof number is equivalent to the Archimedes number but when the density difference between is caused by heat transfer and a thermal expansion coefficient β of the fluid.

Knudsen number Kn

The Knudsen number is the ratio of the mean free path λ_{free} of a molecule in a fluid and a characteristic length scale L (such as the size of an embedded particle),

$$\mathrm{Kn} = \frac{\lambda_{free}}{L}. \tag{3.29}$$

The mean free path is the average distance a molecule travels between collisions with other molecules. For most liquids this distance is relatively small, less than the typical size of the molecule, but in gases the mean free path can be considerably larger. The Knudsen number is a measure for the applicability of continuum scale approaches for solving fluid flow. If Kn $\ll 1$ (in practice < 0.01), the mean free path is sufficiently much smaller than the size of the object for a continuum approach to hold. However, if Kn ≥ 1 the discrete molecular nature of the fluid becomes important, and a continuum approach is no longer appropriate.

Lewis number Le

The Lewis number is the ratio of molecular thermal diffusivity to molecular mass diffusivity,

$$\mathrm{Le} = \frac{\alpha}{D}. \tag{3.30}$$

The Lewis number is relevant in fluid flows where simultaneous heat and mass transfer is taking place. In convective flows, the Lewis number controls the relative thickness of the thermal and mass-transfer boundary layers.

Mach number Ma

The Mach number is the ratio between the velocity of an object moving though a fluid, and the speed of sound in that fluid,

$$\mathrm{Ma} = \frac{V}{c_s}. \tag{3.31}$$

When Ma > 1 shock waves are generated in the fluid, but density waves are also observed for Ma < 1. Generally the effects of finite compressibility scale as Ma^2, meaning that in many cases a Mach number of 0.1 or lower is sufficient to approach "incompressible" behaviour of a fluid.

Nusselt number Nu

The Nusselt number is the ratio of convective to conductive heat transfer normal to a boundary. The conductive component is measured under the same conditions as the heat convection, but for a fictitious stagnant fluid:

$$\mathrm{Nu} = \frac{hL}{k}. \tag{3.32}$$

A Nusselt number close to unity, where convective and conductive heat flows are of similar magnitude, a characteristic of laminar flow. A larger Nusselt number corresponds to more active convection, with turbulent flow typically in the 100-1000 range.

Péclet number Pe

The Péclet number is the ratio of mass convection by flow to mass diffusion of a particle of size L,

$$\mathrm{Pe} = \frac{LV}{D}. \tag{3.33}$$

For $\mathrm{Pe} \ll 1$ the dynamics of the particle is dominated by thermal fluctuations, i.e. Brownian motion. This does not necessarily implicate that the fluid flow can be neglected. At larger time scales, the convection (which is linear in time) may still dominate the diffusion (which scales as $\sqrt{t}$). Conversely, for $\mathrm{Pe} \gg 1$ the dynamics of the particle is dominated by the fluid flow, and thermal fluctuations can be ignored. Note that for mass transfer of a molecular fluid, we have $\mathrm{Pe} = \mathrm{Re} \cdot \mathrm{Sc}$.

In pure fluid flow problems, the Péclet number is sometimes defined differently, as the ratio of convective mass transfer rate to thermal diffusion rate,

$$\mathrm{Pe}_f = \frac{LV}{\alpha}, \tag{3.34}$$

in which case $\mathrm{Pe}_f = \mathrm{Re} \cdot \mathrm{Pr}$.

Prandtl number Pr

The Prandtl number is the ratio of momentum diffusion to thermal diffusion in a fluid,

$$\mathrm{Pr} = \frac{\nu}{\alpha} = \frac{c_P \mu}{k}. \tag{3.35}$$

Note that the Prandtl number is a property of the fluid and the fluid state, but does not depend on a length scale. Generally, in convective flows the Prandtl number controls the relative thickness of the momentum and thermal boundary layers. The Prandtl number of water is around 7, whereas the Prandtl number of air and many other gases is 0.7 to 0.8. The Prandtl number of liquid mercury is 0.015, indicating that in a liquid metal thermal diffusivity is dominant over momentum diffusion.

Reynolds number Re

The Reynolds number is the ratio of inertial to viscous forces. For a fluid moving with velocity V around a particle of size L, the particle Reynolds number is

$$\mathrm{Re} = \frac{LV}{\nu} = \frac{\rho_f LV}{\mu}. \tag{3.36}$$

For a fluid flowing between two plates, through a pipe, or a similar confined geometry of dimension L, the Reynolds number is also given by the above equation. For $\mathrm{Re} \ll 1$ inertial effects can be neglected, meaning that the non-linear term in the Navier-Stokes equation may be neglected. In this so-called Stokes flow regime, the hydrodynamic equations are linear, greatly facilitating theoretical solutions of the fluid flow. For $\mathrm{Re} > 1$ inertial effects become increasingly important. For not too high Reynolds numbers (typically up the order of 100 for flow around a particle and the order of 1000 for flow through a confined geometry) the fluid flow is still laminar, characterised by smooth, constant fluid motion. For even higher Reynolds numbers, turbulent flow occurs, characterised by chaotic eddies, vortices and other flow instabilities.

Schmidt number Sc

The Schmidt number is the ratio of momentum diffusion to molecular mass diffusion,

$$\mathrm{Sc} = \frac{\nu}{D} = \frac{\mu}{\rho_f D}. \tag{3.37}$$

For a fluid flowing past a surface (possibly of a particle), the Schmidt number controls the relative thickness of the hydrodynamic momentum and mass-transfer boundary layers. The Schmidt number of water is approximately 1000, whereas the Schmidt number of air and many other gases is 1.

Sherwood number Sh

The Sherwood number is the equivalent of the Nusselt number for mass transfer problems. It is defined as the ratio of convective mass flux to diffusive mass flux (similar to the Peclet number),

$$\mathrm{Sh} = \frac{KL}{D}, \tag{3.38}$$

where K is the mass transfer coefficient (in m/s).

Weber number

The Weber number measures the relative importance of inertia to surface tension between two fluids,

$$\mathrm{We} = \frac{\rho_f V^2 L}{\gamma}, \tag{3.39}$$

where L is a characteristic size. For example, for two colliding droplets in air, L is the (smallest) droplet diameter. When $\mathrm{We} < 1$ the surface tension dominates and the droplets either coalesce or bounce off each other. When $\mathrm{We} \gg 1$ the inertia of the droplets dominates, resulting in a violent collision with large deformation of the droplets and possible formation of new satellite droplets.

3.4 Microscopic, mesoscopic and macroscopic simulations

We have seen that most particle-based simulation programs share the same features. Still, particle-based simulations can be performed on many different scales with different levels of detail. There are three main types of simulations: microscopic, mesoscopic and macroscopic simulations. Below we explain how to distinguish between these different types of simulations, summarised in Table 3.3. In the following three chapters we will deal with the particular details in detail.

Microscopic simulations

Microscopic simulations are simulations in which all details of the molecular interactions are included. This is the field of molecular dynamics. An important characteristic is that all interactions are conservative, i.e. in terms of potential energies that depend on atom-atom distances, angles, dihedral angles, etcetera. On this molecular scale there is no dissipation of energy, and the motion of the molecules is dominated by thermal fluctuations. Molecular dynamics simulations typically deal with the motion of the order of 10^5 to 10^6 atoms, each of which need to be updated with time steps of the order of a few femtoseconds (1 fs = 10^{-15} s), resulting in system sizes of up to tens of nanometers (1 nm = 10^{-9} m) and total simulation times of up to 100 nanoseconds.

	dissipative forces	thermal flucts.	particle size	time step	gravity	surface tension
microscopic	-	+	$\approx 10^{-10}$ m	$\approx 10^{-15}$ s	-	+
mesoscopic	+	+	$10^{-9} - 10^{-5}$ m	$10^{-13} - 10^{-6}$ s	$\pm$	+
macroscopic	+	-	$\geq 10^{-4}$ m	$\geq 10^{-6}$ s	+	$\pm$

Table 3.3: Distinction between three main types of particle-based simulation methods.

Microscopic systems are too small to feel the effects of gravity. Often diffusive processes (self-diffusion, viscosity, thermal diffusion) and surface properties (surface tension) are of interest. In other words, the Archimedes, Eötvös, Grashof, Peclet, Reynolds and Weber numbers are all very small ($\ll 1$).

Mesoscopic simulations

Mesoscopic simulations are simulations in which the molecular interactions have been lumped into effective interactions between larger assemblies of molecules: they have been coarse-grained. Various simulation methods have been developed in the last decades, such as dissipative particle dynamics, Langevin dynamics, Brownian dynamics, and multiparticle collision dynamics. The effective interactions do not only consist of a conservative part, but also of a dissipative and stochastic part corresponding to thermal fluctuations. So an important characteristic of mesoscopic simulations is that the interactions are dissipative *and* that thermal fluctuations remain important. Systems that fall in this category are also called *soft matter* systems, because small forces are sufficient to change the state of the system. Examples include polymeric and colloidal solutions, oil-and-water emulsions, blood, etcetera. In these lectures we will mainly focus on colloidal solutions, which are suspensions of particles with a diameter between 10 nanometers and 10 micrometers.

Similar to molecular dynamics simulations, mesoscopic simulations typically deal with 10^5 to 10^6 particles, although some methods can deal efficiently with up to 100 times more particles. Because the particles are coarse-grained, larger time steps can be made and larger length scales can be reached than in the case of microscopic simulations. The precise gain depends on the system because, generally, softer interactions and higher frictions allow for larger time steps. System sizes range from tens of nanometers to approximately 100 micrometers, and total simulation times range from 100 nanoseconds to several seconds, depending on the system.

Large mesoscopic systems can already feel the effects of gravity. Both diffusive and convective processes can be important, but convection is usually not dominating strongly. Surface tension is usually still dominating. In other words, the Eötvös and Weber numbers remain very small ($\ll 1$), but the Archimedes, Grashof, Peclet and Reynolds numbers range from very small to about 10.

Macroscopic simulations

Macroscopic particle-based simulations are simulations of large particles that can be seen by the naked eye, i.e. particles of 100 micrometers and larger. An important characteristic is that thermal fluctuations are no longer important, i.e. the self-diffusion of the particles is negligible. The interaction between the particles, usually referred to as a *contact model*, is dissipative. Contrary to the previous two types of simulations, without further perturbations the particles will come to complete rest.[8] Particle-based simulation methods on this scale include Discrete Particle and Discrete Element methods.

Similar to molecular dynamics simulations, macroscopic simulations typically deal with 10^5 to 10^6 particles. The time step depends on the stiffness of the particle interaction. Often the stiffness can be artificially lowered without affecting the results, allowing for time steps as large as 10^{-5} or 10^{-4} s. The resulting system sizes range from several millimeters to a meter, and total simulation times range from seconds to minutes.

For these large particles, the effects of gravity are often dominant. In typical engineering applications, convective processes are of interest and dominate over diffusive processes. Surface tension may or may not be important, depending on the system. In other words, the Eötvös and Weber numbers can range from small (<1) to large (>1), and the Archimedes, Grashof, Peclet and Reynolds numbers are often large (> 1).

[8] Perturbations can be provided by flow of the fluid around the particles, as in a fluidized bed.

CHAPTER 4

THE MICROSCOPIC WORLD

4.1 Chapter objectives

Through the course of this chapter, you will accomplish the following:

- You will learn about force fields in molecular dynamics simulations.
- You will learn how to control the temperature and/or pressure in a particle-based simulation.
- You will learn how to measure structural properties such as the radial distribution function in a particle-based simulation.
- You will learn how to measure dynamic properties such as the self-diffusion coefficient and the viscosity in a particle-based simulation.
- You will learn about limitations of molecular dynamics simulations

4.2 A short introduction to molecular dynamics simulations

Molecular dynamics simulations enable us to calculate structural and thermodynamic properties, as well as dynamical properties, of realistic molecular systems, provided they exclude chemical reactions and other phenomena of a quantum mechanical na-

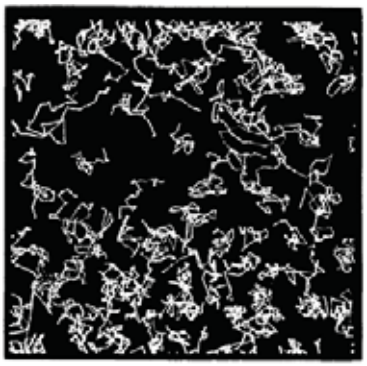

Figure 4.1: The traces of 108 hard-sphere particles with periodic boundary conditions for about 3000 collisions in one of the first molecular dynamics simulations by Alder and Wainwright [2].

ture [4].[1] The first molecular dynamics simulations were performed by Alder and Wainwright in the 1950's [2]. To minimise finite system size effects, they applied periodic boundary conditions, as we discussed in section 2.5. Figure 4.1 shows their result for the trajectories of 108 particles interacting through hard-sphere interactions. Of course real molecules do not interact as hard-spheres. Still, the hard-sphere fluid is a useful concept because it is the simplest representation of repulsive interactions between spherical molecules, with the added advantage that analytical solutions can be obtained for various properties of hard sphere fluids.

For the properties of real molecular fluids, we need more accurate interaction potentials, collectively referred to as *force fields*. The force fields may be obtained from quantum-chemical calculations, but more often the force field parameters are tuned to achieve maximum agreement with experimental thermodynamic data such as the state points (temperature and pressure) where phase transitions take place. A whole industry has emerged that is focusing on developing increasingly accurate force fields. Many open source codes exist that allow us to simulate 10^5 to 10^6 atoms efficiently, such as GROMACS, LAMMPS, NAMD, ESPRESSO, AMBER, DL_POLY, CHARMM, etcetera.

In the following section we will give an example of the popular CHARMM force field. Then we will describe how to simulate different thermodynamic ensembles and measure structural and dynamical properties of the system.

4.3 Molecular force fields

In most molecular force fields, the total potential energy Φ is divided up into terms depending on the coordinates of individual particles, pair, triplets, etc. Bonded interactions within molecules typically consist of a harmonic potential between two bonded atoms, a harmonic bending potential for three consecutively bonded atoms, and a dihedral (torsional) potential for four consecutively bonded atoms. Non-bonded interactions between atoms are usually described by a Lennard-Jones potential, Eq. (2.7),

[1] There is another important technique, called Monte Carlo (MC) simulation, which enables us to efficiently sample the phase space in a specified ensemble. This also yields structural and thermodynamic properties, but no dynamical properties. We will not treat the Monte Carlo technique here.

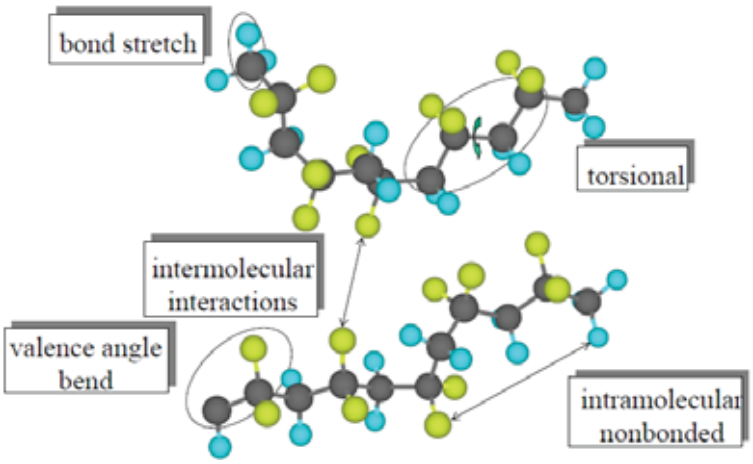

Figure 4.2: In molecular dynamics simulations the interaction potential is usually approximated as a sum of intramolecular bonded pair interactions, valence angle interactions involving three consecutively bonded atoms, dihedral (torsional) interactions between four consecutively bonded atoms, and non-bonded interactions (both intramolecular and intermolecular).

supplemented with Coulombic terms if the atoms are charged. The non-bonded interactions can be further subdivided into intermolecular interactions, i.e. interactions between atoms on different molecules, and intramolecular non-bonded interactions, usually between atoms further than two bonds away on the same molecule. Figure 4.2 shows an example of all these forces in the interaction between two molecules. In the following subsections we will descibe each of the terms in the CHARMM force field:[2]

$$\begin{aligned}\Phi \;=\; & \sum_{\text{bonds}} k_b\,(b-b_0)^2 + \sum_{\text{angles}} k_\theta\,(\theta-\theta_0)^2 + \sum_{\text{dihedrals}} k_\phi\left[1+\cos(n\phi-\delta)\right] \\ & + \sum_{\text{non-bonded}} 4\epsilon_{ij}\left[\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right] + \sum_{\text{charged pairs}} \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}} \qquad (4.1)\end{aligned}$$

Bond stretch interactions

The interaction between two covalently bonded atoms may be approximated as harmonic:

$$\varphi^b(b) = k_b(b-b_0)^2, \qquad (4.2)$$

where $b-b_0$ is the distance from equilibrium that the current pair of bonded atoms has moved, and k_b is the spring stiffness associated with that particular bond.[3]

[2] We have excluded from this description the improper dihedrals which enforce out-of-plane bending, and Urey-Bradley interactions which are cross-terms accounting for angle bending using 1-3 non-bonded interactions.

[3] Note that, confusingly, in the CHARMM force field all harmonic interactions are denoted $k(x-x_0)^2$ instead of the more usual $\frac{1}{2}k(x-x_0)^2$. So k in this force field is actually half the spring stiffness.

Valence angle interactions

The stiffness of angles within a molecule is achieved by including harmonic angle interactions:

$$\varphi^{\theta}(\theta) = k_{\theta}(\theta - \theta_0)^2, \tag{4.3}$$

where $\theta - \theta_0$ is the angle from equilibrium between 3 bonded atoms, and k_θ is the angular stiffness associated with that angle.

Dihedral (torsional) interactions

Torsional stiffness of a molecule is achieved by including dihedral angle interactions:

$$\varphi^{\phi}(\phi) = k_{\phi}\left[1 + \cos(n\phi - \delta)\right], \tag{4.4}$$

where ϕ is the dihedral angle between 4 bonded atoms (the angle between the plane of atoms 1,2,3 and the plane of atoms 2,3,4), n is the multiplicity of the dihedral, and δ is the phase shift (to control the location of the equilibrium dihedrals).

Non-covalently bonded interactions

Non-covalently bonded atoms usually attract each other through Van der Waals interactions (caused by induced dipole-interactions) that scale as r^{-6}. When non-bonded atoms come too close together they will repel each other because of the Pauli exclusion principle. A convenient way to summarize these two effects is through the Lennard-Jones potential, as introduced in section 2.3, which we repeat here for completeness. For two particles i and j a distance r_{ij} apart:

$$\varphi^{LJ}(r_{ij}) = 4\epsilon_{ij}\left\{\left(\frac{\sigma_{ij}}{r_{ij}}\right)^{12} - \left(\frac{\sigma_{ij}}{r_{ij}}\right)^{6}\right\}. \tag{4.5}$$

The parameter ϵ_{ij} is the depth of the interaction well, and σ_{ij} is the effective diameter for the pair interaction, both of which depend on the atom type, and even chemical environment, of both i and j. Often the following simplified "mixing rule" is used,

$$\sigma_{ij} = \frac{1}{2}\left(\sigma_i + \sigma_j\right), \tag{4.6}$$

$$\epsilon_{ij} = \sqrt{\epsilon_i \epsilon_j}, \tag{4.7}$$

where σ_i is the effective diameter and ϵ_i the effective dispersion energy of atom i, and similarly for atom j.

Charge interactions

When dealing with ions, we clearly need to take into account the Coulombic charge interactions. For two ions with charge q_i and q_j we have

$$\varphi^C(r_{ij}) = \frac{1}{4\pi\epsilon_0}\frac{q_i q_j}{r_{ij}}, \tag{4.8}$$

where $\epsilon_0 = 8.854 \times 10^{-12}$ F/m is the permittivity of vacuum.

Besides ions, charge interactions are often also important in molecular interactions. Although the distribution of negative electron charges largely cancels out the positive charge of the atomic nuclei, the cancellation is often not perfect in molecules, and a small net charge remains.

4.4 Controlling temperature and pressure: thermostats and barostats

The force field leads to a force on each of the atoms. This force may be used to update the velocities and positions of the atoms to the next time step. Proceeding in this fashion, molecular dynamics simulations are in principle energy conserving: the sum of kinetic and potential energy is conserved. Experimentally, it is very difficult to control the energy of a system. In most realistic cases we control the temperature. Similarly, we may wish to control the pressure of a molecular system instead of the system volume. In this section we will describe how this may be achieved in a molecular dynamics simulation. For this, we will need a short recap of statistical thermodynamics.

Short recap of statistical thermodynamics

Consider a system composed of N particles in a volume V. According to quantum mechanics this system may find itself in one of a countable number of states each with its own energy E_n, where n is the index of the particular state. Now suppose that we fix the energy of the system to the value U. Then there are $\Omega(U)$ states for which the energy is equal to U, where Ω is the degeneracy of energy U. Although each of these states has the same energy, there may be other observables A of interest having different values A_n in different states. According to statistical physics a macroscopic measurement of the observable A will yield the ensemble average[4]

$$\langle A \rangle = \sum_n P_n A_n \tag{4.9}$$

Where P_n is the probability to observe state n:

$$P_n = \begin{cases} \frac{1}{\Omega(U)} & E_n = U \\ 0 & E_n \neq U \end{cases} \tag{4.10}$$

[4]We will consistently denote the ensemble average of an observable A as $\langle A \rangle$ and the time-average of A as $\langle A \rangle_T$. The ergodicity hypothesis states that for equilibrium systems $\langle A \rangle = \langle A \rangle_T$ [45].

So, it is assumed that each of the states with energy U has equal probability $1/\Omega(U)$.

In many cases, instead of fixing the energy we fix the temperature T. The way to do this is to put the system in contact with a huge second system with which it may exchange energy, and which is called a thermostat. By isolating the totality of system plus thermostat we may apply the rules just stated and calculate the average of any observable, in particular of any observable defined in terms of the particles composing the system and independent of the particles composing the thermostat. By doing so, we again find Eq. (4.9), but now with

$$P_n = \frac{1}{Q}\exp\{-\beta E_n\} \tag{4.11}$$

$$Q = \sum_n \exp\{-\beta E_n\} \tag{4.12}$$

$$\beta = \frac{1}{k_B T} \tag{4.13}$$

where k_B is Boltzmann's consant and Q is referred to as the partition function. Notice that in this case the energy is not fixed, but that it is assumed that a macroscopic measurement yields the average energy $U = \sum_n P_n E_n$

One way to envision a macroscopic measurement is to imagine being given a huge number of similar systems in different states distributed according to the probabilities given in Eq. (4.10) or (4.11). Such collections of systems are traditionally called ensembles. Eq. (4.10) represents the so-called micro-canonical or (N,V,U) ensemble and Eq. (4.11) the canonical or (N,V,T) ensemble. From the above it is clear that in the (N,V,T) ensemble the energy is a fluctuating quantity. The width of the distribution of energies is conveniently measured by $\sigma_E = \sqrt{\langle E^2\rangle - \langle E\rangle^2}$. From statistical physics we know that

$$\frac{\sigma_E}{\langle E\rangle} \sim \frac{1}{\sqrt{N}} \tag{4.14}$$

Evidently, for large enough systems all relative fluctuations are insignificant and all ensembles yield the same results. This is called the *thermodynamic equivalence* for large systems.

It is a known fact that for large systems the degeneracy Ω (in a microcanonical system) scales as follows:

$$\Omega(N,V,U) = \omega\left(\frac{V}{N},\frac{U}{N}\right)^N O(N)\,. \tag{4.15}$$

Here $O(N)$ represents a factor which growths with N at most as fast as N itself. Therefore it is convenient to define

$$S = k_B \ln \Omega \tag{4.16}$$

which, for large enough systems, is proportional to the system size. According to statistical physics it has all the properties of the entropy. Since S is given in terms of its

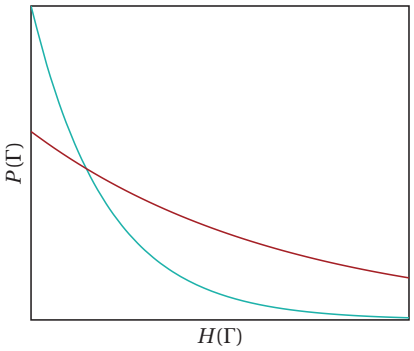

Figure 4.3: The classicale probability of encountering a system near a certain point Γ in phase space decreases exponentially with the associated Hamiltonian (energy) H (green). High values of H are sampled with higher probability at higher temperature (red).

characteristic variables, N, V and U, all thermodynamic information can be extracted from it. Applying this equation to a system in a thermostat, we find

$$A = -kT \ln Q \tag{4.17}$$

where A is the free energy.

Although for convenience the above discussion has been given in terms of discrete states like they occur in quantum mechanical treatments of confined systems, it is equally possible to use the language of classical mechanics. Taking the classical limit (i.e. excluding very low temperatures) turns the degeneracy Ω into a density-of-states

$$\Omega = \frac{1}{h^{3N} N!} \int d\Gamma \, \delta \left(H(\Gamma) - U \right) \tag{4.18}$$

where h is Planck's constant, $\Gamma = \left(r^{3N}, p^{3N} \right)$ is a point in phase space and H is the Hamiltonian (potential plus kinetic energy) of the system. The above equations indicates that, in the classical limit, each point in phase space on a hypersurface of energy U is equally likely to occur. The canonical partition function Q becomes

$$Q = \frac{1}{h^{3N} N!} \int d\Gamma \exp \left\{ -\beta H(\Gamma) \right\}, \tag{4.19}$$

where the classical probability of observing the system near a certain point (Γ) in phase space is given by the **Boltzmann distribution** (Figure 4.3):

$$P(\Gamma) \mathrm{d}\Gamma \sim \exp \left\{ -\beta H(\Gamma) \right\} \mathrm{d}\Gamma. \tag{4.20}$$

Thermostats

In all relevant examples that we will encounter, the Hamiltonian splits into a potential energy that only depends on the particle positions, and a kinetic energy that only depends on the particle velocities, i.e.

$$H(\Gamma) = \Phi(\mathbf{r}_1, \ldots, \mathbf{r}_N) + \sum_{i=1}^{N} \frac{1}{2} m_i v_i^2 \tag{4.21}$$

If we combine this with Eq. (4.20), we find that the classical probability factorises into a probability for the positions and a probability for the velocities:

$$P(\mathbf{r}_1,\ldots,\mathbf{r}_N,\mathbf{v}_1,\ldots,\mathbf{v}_N) = P^r(\mathbf{r}_1,\ldots,\mathbf{r}_N)P^v(\mathbf{v}_1,\ldots,\mathbf{v}_N). \tag{4.22}$$

where the velocity probability density P^v is given by

$$P^v(\mathbf{v}_1,\ldots,\mathbf{v}_N) \sim \exp\left\{-\beta\sum_{i=1}^{N}\frac{1}{2}m_i v_i^2\right\} = \prod_{i=1}^{N}\exp\left\{-\frac{m_i v_i^2}{2k_B T}\right\}. \tag{4.23}$$

This is exactly the **Maxwell-Boltzmann distribution**, which we have used several times before! The mean square velocity[5] is therefore a direct function of the system temperature:

$$\left\langle v_i^2\right\rangle = \frac{k_B T}{m_i}. \tag{4.24}$$

We can invert this relationship to determine the instantaneous temperature of a system from the particle velocities:

$$T^{meas} = \frac{1}{N_{free}k_B}\sum_i m_i v_i^2, \tag{4.25}$$

where N_{free} is the number of degrees of freedom. If we use periodic boundary conditions and have removed the initial centre-of-mass velocity of the system, the number of degree of freedom in d dimensions is $N_{free} = d(N-1)$; otherwise $N_{free} = dN$. The instantaneous temperature T^{meas} of the system is not necessarily equal to the thermodynamic temperature. Only the long time average, or ensemble average, $\langle T^{meas}\rangle$ of a simulation is equal to the thermodynamic temperature T of the system.

Temperature constraining through direct velocity scaling

An obvious way to control the thermodynamic temperature of the system is velocity scaling. If the temperature of the system at time t is $T^{meas}(t)$ and the velocities are multiplied by a factor λ,

$$\mathbf{v}_i \mapsto \lambda\mathbf{v}_i, \tag{4.26}$$

then the associated temperature change can be calculated as

$$\begin{aligned}\Delta T &= \frac{1}{N_{free}k_B}\sum_i m_i(\lambda v_i)^2 - \frac{1}{N_{free}k_B}\sum_i m_i v_i^2 \\ &= \left(\lambda^2-1\right)T^{meas}(t).\end{aligned} \tag{4.27}$$

[5] Note that statistical thermodynamics applies to equilibrium states, and that therefore we exclude average flow fields from our description.

The simplest way to control the temperature is thus to multiply the velocities at each time step by a factor

$$\lambda = \sqrt{\frac{T_0}{T^{meas}}} \qquad \text{(direct scaling)}, \tag{4.28}$$

where T^0 is the desired temperature (check this for yourself!). This direct velocity scaling leads to a strict constraint on the kinetic energy.

Velocity scaling: the Berendsen thermostat

One problem with direct velocity scaling is that is does not allow fluctuations in the instantaneous temperature T^{meas} which should be present in the canonical ensemble. A weaker formulation of the velocity scaling approach is the Berendsen thermostat. To maintain the temperature, the system is coupled to an external heat bath of fixed temperature T_0. The velocities are scaled at each step, such that the rate of change of temperature is proportional to the difference in temperature:

$$\frac{\mathrm{d}T(t)}{\mathrm{d}t} = \frac{1}{\tau}\left(T_0 - T(t)\right), \tag{4.29}$$

where τ is the coupling parameter which determines how tightly the heat bath and the system are coupled together. This method gives an exponential decay of the system towards the desired temperature. The required change in temperature between successive times steps Δt is:

$$\Delta t = \frac{\Delta t}{\tau}\left(T_0 - T(t)\right). \tag{4.30}$$

Thus, the scaling factor for the velocities is

$$\lambda^2 = 1 + \frac{\Delta t}{\tau}\left\{\frac{T_0}{T(t)} - 1\right\} \qquad \text{(Berendsen)}. \tag{4.31}$$

The following pseudo-code shows how the velocies could be rescaled using the Berendsen thermostat. Note that for efficiency, it would be better to make this rescaling a direct part of the discretised update of the velocity.

```
routine berendsen_thermostat
Ekin = 0
Nfree = 3*(N-1)
do i=1,N
  Ekin = Ekin+m(i)*(vx(i)*vx(i)+vy(i)*vy(i)+vz(i)*vz(i))
enddo
Tmeas = Ekin/(Nfree*kboltz)
lambda = sqrt(1+dt/tau*(T0/Tmeas-1))
do i=1,N
```

```
  vx(i) = lambda*vx(i)
  vy(i) = lambda*vy(i)
  vz(i) = lambda*vz(i)
enddo
end routine
```

In practice, τ is used as an empirical parameter to adjust the strength of the coupling. Its value has to be chosen with care. In the limit $\tau \to \infty$ the Berendsen thermostat is inactive and the run is sampling a microcanonical (constant energy) ensemble. The temperature fluctuations will grow until they reach the appropriate value of a microcanonical ensemble. However, they will never reach the appropriate value for a canonical ensemble. On the other hand, too small values of τ will cause unrealistically low instantaneous temperature fluctuations. If τ is chosen the same as the timestep Δt, the Berendsen thermostat is nothing else than the direct velocity scaling thermostat. Values of $\tau \approx 0.1$ ps are typically used in molecular dynamics simulations of condensed-phase systems. We should realise, however, that the ensemble generated when using the Berendsen thermostat is not a canonical ensemble.

Adding an external variable: the Nosé-Hoover thermostat

If it is important to sample a correct canonical ensemble, the best way to control the temperature in a molecular dynamics simulation is to let the system exchange energy with an external reservoir represented by a new coordinate with associated mass Q and velocity. The magnitude of the mass Q determines the coupling between the system and the reservoir, and so influences the (instantaneous) temperature fluctuations. This leads to the so-called Nosé equations of motion, which are rather complex to write down here. However, Nosé and Hoover showed that the equations may also be expressed in a more intuitive and simple way as:

$$\frac{\mathrm{d}^2\mathbf{r}_i}{\mathrm{d}t^2} = \frac{\mathbf{F}_i}{m_i} - \gamma(t)\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}, \tag{4.32}$$

$$\frac{\mathrm{d}\gamma}{\mathrm{d}t} = \frac{1}{Q}\left\{\sum_{i=1}^{N} m_i v_i^2 - N_{free}k_B T_0\right\}. \tag{4.33}$$

Note that γ acts like time-dependent friction caused by coupling with the reservoir. This friction is also a dynamical variable, which is updated with a rate controlled by the fictitious mass Q. Care must be taken in choosing Q. On the one hand, too large values of Q (loose coupling) may cause a poor temperature control (if $Q \to \infty$ the run is sampling the microcanonical ensemble). Although any finite (positive) Q is sufficient to guarantee in principle the generation of a canonical ensemble, if Q is too large, the canonical distribution will only be obtained after very long simulation times. On the other hand, too small values (tight coupling) may cause high-frequency temperature oscillations. The variable γ may oscillate at a very high frequency, it will tend to be

off-resonance with the characteristic frequencies of the real system, and effectively decouple from the physical degrees of freedom (slow exchange of kinetic energy). Usually it requires a few trial-and-error guesses to find a correct fictitious mass Q.

Barostats

Up to this point we have assumed that our system is evolving in a constant volume. Many processes in our daily lives take place under atmospheric pressure, rather than constant volume. It is possible to control the pressure in molecular dynamics simulations by dynamically rescaling the box dimensions. To understand how, we first derive a microscopic expression for the pressure.

Virial expression for the pressure

Consider a system of N atoms that is developing in a finite space. Let us introduce a function, called the *Clausius virial function*:

$$W^{tot}(r^{3N}) = \sum_{i=1}^{N} \mathbf{r}_i \cdot \mathbf{F}_i^{tot}, \tag{4.34}$$

where $\mathbf{F}_i^{tot}$ is the total force acting on atom i, both due to internal forces with other atoms and due to external forces with confining walls. Averaging over the molecular dynamics trajectory, and using Newton's law, we find

$$\begin{aligned} \langle W^{tot} \rangle &= \lim_{T\to\infty} \frac{1}{T} \int_0^T \sum_{i=1}^{N} \mathbf{r}_i(t) \cdot m_i \frac{\mathrm{d}^2 \mathbf{r}_i}{\mathrm{d}t^2}(t) \mathrm{d}t \\ &= \lim_{T\to\infty} \sum_{i=1}^{N} m_i \frac{\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(T) \cdot \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(T) - \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(0) \cdot \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(0)}{T} \\ &\quad - \lim_{T\to\infty} \frac{1}{T} \int_0^T \sum_{i=1}^{N} m_i \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(t) \cdot \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(t) \mathrm{d}t \end{aligned} \tag{4.35}$$

where we have integrated by parts. If the system is localized in a finite region and particles are not accelerated to infinity, then the first term of the second line is zero. We then recognize that the expectation value of the Clausius virial function is related to the temperature of the system:

$$\langle W^{tot} \rangle = -\lim_{T\to\infty} \int_0^T \sum_{i=1}^{N} m_i \left| \frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t}(t) \right|^2 \mathrm{d}t = -2 \langle E_{kin} \rangle = -N_{free} k_B T \tag{4.36}$$

Writing the total force on atom i as $\mathbf{F}_i^{tot} = \mathbf{F}_i^{int} + \mathbf{F}_i^{ext}$, we can write the total virial function as a sum of internal and external virials:

$$\langle W^{tot} \rangle = \left\langle \sum_i \mathbf{r}_i \cdot \mathbf{F}_i^{int} \right\rangle + \langle W^{ext} \rangle = -N_{free} k_B T. \tag{4.37}$$

Now, how do we connect this to the macroscopic pressure? The macroscopic pressure P is the normal force per unit area exerted by walls surrounding our system. For simplicity, we consider a rectangular container with sides L_x, L_y and L_z, and the coordinate origin on one of its corners. Then the force exerted on the atoms by the right wall at $x = L_x$ is given by $F_x^{ext} = -PL_yL_z$. Similarly, the force exerted on the atoms by the back wall at $y = L_y$ is given by $F_y^{ext} = -PL_xL_z$, and the force exerted on the atoms by the top wall at $z = L_z$ is given by $F_z^{ext} = -PL_xL_y$. The forces exerted by the left, front and bottom walls are irrelevant for the virial function because atoms influenced by these walls have x, y and z-positions very close to zero, respectively. Therefore, the external virial is given by

$$\left\langle W^{ext} \right\rangle = L_x\left(-PL_yL_z\right) + L_y\left(-PL_xL_z\right) + L_z\left(-PL_xL_y\right) = -3PV. \tag{4.38}$$

Combining this with the total virial equation, we arrive at the virial expression for the pressure P:

$$P = \frac{Nk_BT}{V} + \frac{1}{3V}\left\langle \sum_{i=1}^{N} \mathbf{r}_i \cdot \mathbf{F}_i^{int} \right\rangle, \tag{4.39}$$

where we have approximated the number of degrees of freedom by $3N$ (generally $N \gg 1$). The first term is the kinetic contribution to the pressure (the pressure of an ideal gas), and the second term is often referred to as the virial part of the pressure.

The virial expresion is very important because it allows us to calculate the pressure of a system entirely in terms of particle coordinates and internal forces, without reference to an actual wall. For pairwise interactions:[6]

$$\begin{aligned} \sum_i \mathbf{r}_i \cdot \mathbf{F}_i^{int} &= \sum_i \sum_{j\neq i} \mathbf{r}_i \cdot \mathbf{F}_{ij} = \frac{1}{2}\sum_i \sum_{j\neq i} \left(\mathbf{r}_i \cdot \mathbf{F}_{ij} + \mathbf{r}_j \cdot \mathbf{F}_{ji}\right) \\ &= \frac{1}{2}\sum_i \sum_{j\neq i} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij} = \sum_i \sum_{j>i} \mathbf{r}_{ij} \cdot \mathbf{F}_{ij}. \end{aligned} \tag{4.40}$$

In other words, for atoms interacting through a pairwise interaction potential $\varphi(r)$, the pressure is given by:

$$P = \frac{Nk_BT}{V} - \frac{1}{3V}\left\langle \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} r_{ij} \left.\frac{\mathrm{d}\varphi(r)}{\mathrm{d}r}\right|_{r_{ij}} \right\rangle \tag{4.41}$$

Note that the virial expression also allows us to calculate the pressure in a periodic system, in which no walls are present at all. In that case V is the volume of one of the periodic cells, and the pair sum should run over each minimum image pair exactly once.

[6] More generally, for any interaction for which the sum of forces on groups of particles is zero, the virial pressure is obtained by summing $(\mathbf{r}_i - \mathbf{r}_{ref}) \cdot \mathbf{F}_i$ over this group, with a suitably chosen reference position $\mathbf{r}_{ref}$.

Box size scaling: the Berendsen barostat

Having obtained a microscopic expression for the pressure, we may control it much in the same way as we control the temperature, in this case by rescaling the box volume and optionally the fractional atom coordinates $(x_i/L_x, y_i/Ly, z_i/Lz)$ (preferred for periodic boundary conditions). Similarly to the Berendsen thermostat, we may also weakly couple the system to an external pressure reservoir as follows:

$$\frac{\mathrm{d}P(t)}{\mathrm{d}t} = \frac{1}{\tau_P}\left(P_0 - P(t)\right), \tag{4.42}$$

where P_0 is the pressure of the external pressure reservoir and τ_P is the response time of the barostat. This is achieved by scaling the volume of the box by μ^3 and the positions by μ every time step, where

$$\mu = \left[1 - \frac{\Delta t}{\tau_P}\kappa\left(P(t) - P_0\right)\right]^{1/3} \qquad \text{(Berendsen)}. \tag{4.43}$$

Here $\kappa = -1/V(\partial V/\partial P)$ is the compressibility of the system. The following pseudocode shows how the positions could be rescaled using the Berendsen barostat, assuming that a thermostat is also used to reach a temperature T_0. Note that for efficiency it would be better to use the neighbourlist and to make this rescaling a part of the discretised update of the position.

```
routine berendsen_barostat
W = 0
do i=1,N-1
  do j=i+1,N
    ...calculate pair distance rij and pair force Fij...
    W = W + rij*Fij
  enddo
enddo
Pmeas = (N*kboltz*T0 + W/3) / (Lx*Ly*Lz)
mu = (1-dt/tau_p*kappa*(Pmeas-P0))^(1/3)
Lx = Lx*mu
Ly = Ly*mu
Lz = Lz*mu
do i=1,N
  x(i) = x(i)*mu
  y(i) = y(i)*mu
  z(i) = z(i)*mu
enddo
end routine
```

Often the compressibility κ is not known beforehand, and the relaxation time τ_P is simply an empirical paramter to tune the strength of the pressure coupling, controlling the

allowed pressure and volume fluctuations. In such a case, the combination κ/τ_P (in units 1/(Pa.s)) can be treated as one single parameter of the Berendsen thermostat. In the limit $\kappa/\tau_p \to 0$ the Berendsen barostat is inactive and the run is sampling a microcanonical (constant energy) ensemble. The pressure fluctuations will grow until they reach the appropriate value of a microcanonical ensemble. On the other hand, if τ_p is chosen the same as the time step Δt (and a realistic κ is used), the Berendsen barostat leads to a constant pressure $P(t) = P_0$, without any pressure fluctuations. This is also not desirable. Optimal values of κ/τ lie in between these two extremes.

If correct fluctuations of the pressure are important (e.g. when one is interested in thermodynamic properties of the system), a barostat can be implemented, similarly to the Nosé-Hoover thermostat, by extending the system with additional coordinates. This so-called Parrinello-Rahman barostat is beyond the scope of these lectures.

4.5 Measuring structural properties

The positions and orientations of atoms are not completely random because the total potential energy Φ contains terms which depend on the relative positions and orientations of two or more molecules. From Eqs. (4.20) and (4.21) we learn that the positional probability density in the canonical ensemble is given by

$$P^r(r^{3N}) = \frac{1}{Z}\exp\left(-\frac{\Phi(r^{3N})}{k_B T}\right), \tag{4.44}$$

where Z is a normalisation constant, also known as the configuration integral. In other words, the interactions between the molecules in a liquid or gas cause correlations in their positions. During the execution of a molecular dynamics simulation, we can measure these structural correlations in the system.

Here we will focus on the two best-known structural correlations, namely the radial distribution function and the static scattering function.

The radial distribution function $g(r)$

The aim of almost all modern theories of liquids is to calculate the radial distribution function by means of statistical thermodynamical reasoning. Alternatively, the radial distribution function can be measured directly in particle-based computer simulations. We will discuss its use in calculating the energy, compressibility and pressure of a fluid, with a particular application to a hard sphere fluid.

Definition of the radial distribution function

Imagine we have placed ourselves on a certain molecule in a liquid or gas (Fig. 4.4). Now let us count the number of molecules in a spherical shell of thickness dr at a distance r, i.e. we count the number of molecules within a distance between r and

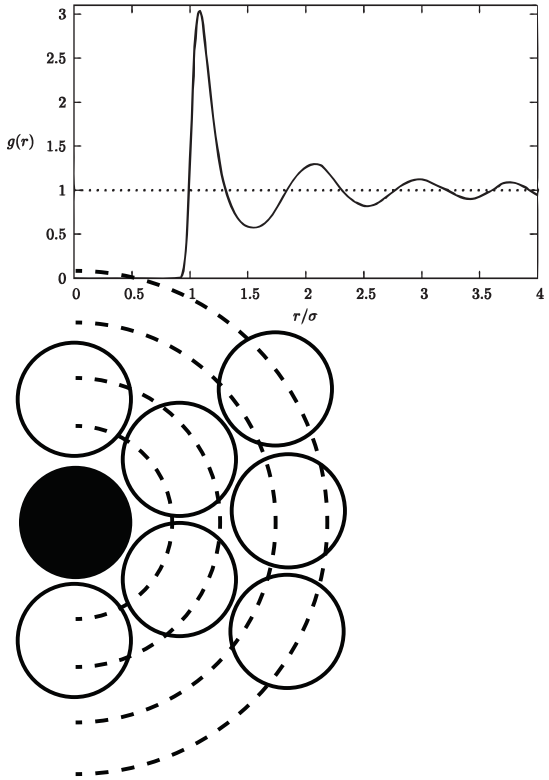

Figure 4.4: A typical radial distribution function in a liquid of spherical molecules with diameter σ. The radial distribution function $g(r)$ measures the local number density of particles at a distance r from a given particle (black circle), relative to the average number density $\rho^{\#} = N/V$.

$r + \mathrm{d}r$. If r is very large the measured number of molecules will be equal to the volume of the spherical shell times the number density $\rho^{\#} = N/V$, so equal to $4\pi r^2 \mathrm{d}r N/V$. At distances smaller than the diameter of the molecules we will find no molecules at all. We now define the radial distribution function $g(r)$ by equating the number of molecules in the spherical shell of thickness $\mathrm{d}r$ at a distance r to

$$4\pi r^2 \frac{N}{V} g(r) \mathrm{d}r. \tag{4.45}$$

According to our remarks above, $g(\infty) = 1$ and $g(0) = 0$. A typical $g(r)$ is given in Fig. (4.4). We see that $g(r) = 0$ when r is smaller than the molecular diameter σ. The first peak is caused by the attractive part of the potential; at distances where the potential has its minimum there are more particles than average. Consequently at distances less than σ further away there are less particles than average.

Measuring the radial distribution function in a molecular dynamics code is accomplished by initialising with `initialise_gr`, calling the routine `update_gr` at regular intervals during the simulation, and finalising the calculation with `finalise_gr`. We divide the range from $r = 0$ to half the box length into `nbin` bins of width `dbin`.

```
routine initialise_gr
grcount = 0                        !number of calls to update_gr
nbin    = 100                      !number of bins
dbin    = 0.5*min(Lx,Ly,Lz)/nbin !bin width
```

```
grbin(:) = 0
end routine

routine update_gr
do i = 1,N-1
  do j = i+1,N
    ...calculate pair distance rij...
    ibin = int(rij/dbin)
    if (ibin < nbin) grbin(ibin) = grbin(ibin)+2
  enddo
enddo
grcount = grcount+1
end routine
```

The bin index is calculated for each pair distance r_{ij}. The bin is updated by 2 because i is a neighbour of j and j is also a neighbour of i, while we process each unique pair i and j only once.

```
routine finalise_gr
rho = (N-1)/(Lx*Ly*Lz)          !number of other particles is N-1
do ibin = 0,nbin-1
  volume = 4/3*pi*((ibin+1)^3-(ibin)^3)*dr^3
  grbin(ibin) = grbin(ibin)/(rho*volume*grcount*N)
  print (ibin+0.5)*dr, grbin(ibin)
enddo
end routine
```

When creating the output we should normalise our results properly. The variable `volume` is equal to the volume of the spherical shell between `ibin*dr` and `(ibin+1)*dr`. The number of particles in an ideal gas one would expect in that volume is $(N-1)/V$ times the spherical shell volume (we must subtract the one particle in the origin). Additionally we need to divide by the number of particles N, because we essentially added the radial distribution functions of N particles, and the number of calls to `update_gr`.

Statistical formulas for $g(r)$

Integrating the probability density for a configuration of N spherical particles, Eq. (4.44), over the coordinates of all particles except the first two, we find

$$P_{12}(\mathbf{r}_1,\mathbf{r}_2) = \frac{1}{Z}\int \mathrm{d}^3 r_3 \ldots \int \mathrm{d}^3 r_N \exp\left(-\frac{\Phi(r^{3N})}{k_B T}\right), \tag{4.46}$$

where $P_{12}(\mathbf{r}_1,\mathbf{r}_2)$ is the probability density to have particle 1 at $\mathbf{r}_1$ and particle 2 at $\mathbf{r}_2$. For convenience of notation we write

$$P_{12}(\mathbf{r},\mathbf{r}') = \frac{1}{Z}\int \mathrm{d}^3 r_3 \ldots \int \mathrm{d}^3 r_N \exp\left(-\frac{\Phi(r^{3N})}{k_B T}\right)\Bigg|_{\mathbf{r}_1=\mathbf{r},\mathbf{r}_2=\mathbf{r}'}. \tag{4.47}$$

Because all particles are equal, this is equal to the probability density $P_{1j}(\mathbf{r},\mathbf{r}')$ of having particle 1 at $\mathbf{r}$ and particle j at $\mathbf{r}'$. The probability density of having particle 1 at $\mathbf{r}$ and *any* other particle at $\mathbf{r}'$ equals

$$\sum_{j\neq 1} P_{1j}(\mathbf{r},\mathbf{r}') = (N-1)P_{12}(\mathbf{r},\mathbf{r}') \tag{4.48}$$

$$\frac{1}{V}\rho^{\#} g(|\mathbf{r}-\mathbf{r}'|) = (N-1)P_{12}(\mathbf{r},\mathbf{r}') \tag{4.49}$$

This is equal to the probability density of having particle 1 at $\mathbf{r}$, which is simply $1/V$, times the conditional density at $\mathbf{r}'$, which is $\rho^{\#} g(|\mathbf{r}-\mathbf{r}'|)$. Multiplying by N we get

$$\left(\rho^{\#}\right)^2 g(|\mathbf{r}-\mathbf{r}'|) = N(N-1)P_{12}(\mathbf{r},\mathbf{r}'). \tag{4.50}$$

Once we know $g(r)$, we can derive all non-entropic thermodynamic properties.

Relation between the radial distribution function and energy

The simplest is the energy:

$$U = U^{\text{int}} + \frac{3}{2}Nk_BT + \frac{1}{2}N\frac{N}{V}\int_0^\infty \mathrm{d}r 4\pi r^2 g(r)\varphi(r). \tag{4.51}$$

The first term originates from the internal energies of the molecules, the second from the translations, and the third from the interactions. The average total potential energy equals $\frac{1}{2}N$ times the average interaction of one particular molecule with all others; the factor $\frac{1}{2}$ serves to avoid double counting. The contribution of all particles in a spherical shell of thickness $\mathrm{d}r$ at a distance r to the average interaction of one particular particle with all others is $4\pi r^2 \mathrm{d}r(N/V)g(r)\varphi(r)$. Integration finally yields Eq. (4.51).

Relation between the radial distribution function and compressibility

The isothermal compressibility κ_T is defined as:

$$\kappa_T \equiv -\frac{1}{V}\left(\frac{\partial V}{\partial P}\right)_{T,N} \tag{4.52}$$

From thermodynamics it is known that κ_T can be linked to spontaneous fluctuations in the number of particles in an open volume V, see Fig. 4.5:

$$\langle N\rangle \rho^{\#} k_B T\kappa_T = \left\langle (N-\langle N\rangle)^2\right\rangle = \left\langle N^2\right\rangle - \langle N\rangle^2, \tag{4.53}$$

where the pointy brackets indicate a long time average or an average over many independent configurations commensurate with the thermodynamic conditions (in this case constant temperature T and volume V). From Eq. (4.50) we obtain (where $r_{12} = |\mathbf{r}_1-\mathbf{r}_2|$):

$$\int_V \mathrm{d}^3 r_1 \int_V \mathrm{d}^3 r_2 \left(\rho^{\#}\right)^2 g(r_{12}) = \langle N(N-1)\rangle = \left\langle N^2\right\rangle - \langle N\rangle. \tag{4.54}$$

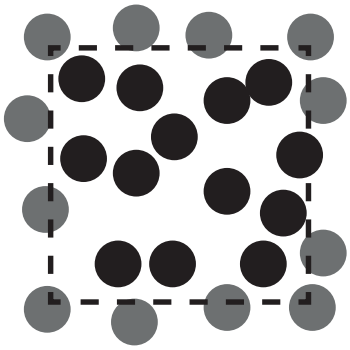

Figure 4.5: The compressibility of a fluid is a measure for the magnitude of spontaneous fluctuations in the number of particles (black circles) in an open volume V (indicated by a dashed line).

We can use this to link the compressibility to the radial distribution function:

$$\begin{aligned}\langle N\rangle \rho^{\#} k_B T \kappa_T &= \rho^{\#} \int_V \mathrm{d}^3 r_1 \rho^{\#} \int_V \mathrm{d}^3 r_2 g(r_{12}) + \langle N\rangle - \rho^{\#} \int_V \mathrm{d}^3 r_1 \rho^{\#} \int_V \mathrm{d}^3 r_2 \\ &= \rho^{\#} \int_V \mathrm{d}^3 r_1 \rho^{\#} \int_V \mathrm{d}^3 r_2 \big(g(r_{12}) - 1\big) + \langle N\rangle \\ &= \rho^{\#} \int_V \mathrm{d}^3 r_1 \rho^{\#} \int_{\mathbb{R}^3} \mathrm{d}^3 r \big(g(r) - 1\big) + \langle N\rangle \end{aligned} \tag{4.55}$$

Dividing by $\langle N\rangle$ we find

$$\rho^{\#} k_B T \kappa_T = 1 + \rho^{\#} \int_{\mathbb{R}^3} \mathrm{d}^3 r \big(g(r) - 1\big). \tag{4.56}$$

This so-called compressibility equation shows that the compressibility of a fluid is intimately connected to the radial distribution function of its constituent molecules.

Relation between the radial distribution function and pressure

We will now consider the pressure of a fluid. We can rewrite the virial expression for the pressure, Eq. (4.39), as:

$$P = \rho^{\#} k_B T - \frac{1}{6} \big(\rho^{\#}\big)^2 \int_0^{\infty} \mathrm{d}r 4\pi r^2 g(r) r \frac{\mathrm{d}\varphi}{\mathrm{d}r}(r). \tag{4.57}$$

Try to derive this equation yourself.

The above equation is valid for a fluid of any density. If the density of the fluid is not too high, correlations between three or more particles may be ignored, in which case Eq. (4.44) tells us that the radial distribution function is given by

$$g(r) \approx \exp\big(-\beta\varphi(r)\big), \tag{4.58}$$

where $\varphi(r)$ is the pair interaction potential. Inserting this into Eq. (4.57) we find after some manipulation:

$$\begin{aligned} P &\approx \rho^{\#} k_B T - \frac{1}{6}\left(\rho^{\#}\right)^2 \int_0^\infty \mathrm{d}r 4\pi r^3 \exp\left(-\beta\varphi(r)\right) \frac{\mathrm{d}\varphi}{\mathrm{d}r}(r) \\ &= \rho^{\#} k_B T + \frac{1}{6}\left(\rho^{\#}\right)^2 \int_0^\infty \mathrm{d}r 4\pi r^3 k_B T \frac{\partial}{\partial r}\left[\exp\left(-\beta\varphi(r)\right) - 1\right] \\ &= \rho^{\#} k_B T + \frac{2\pi}{3}\left(\rho^{\#}\right)^2 k_B T \left\{ r^3 \left[\exp\left\{-\beta\varphi(r)\right\} - 1\right]\Big|_0^\infty - \int_0^\infty \mathrm{d}r 3r^2 \left[\exp\left\{-\beta\varphi(r)\right\} - 1\right] \right\} \\ &= \rho^{\#} k_B T \left\{ 1 - 2\pi\rho^{\#} \int_0^\infty \mathrm{d}r\, r^2 \left[\exp\left\{-\beta\varphi(r)\right\} - 1\right] \right\}. \end{aligned} \tag{4.59}$$

In the second line we have used the chain rule of differentiation, in the third line we integrated by parts, and in the fourth line we used the fact that $\varphi(r \to \infty) = 0$ to get rid of the first term.

In the classical chemical and physical literature, the pressure of a fluid is often expressed as an expansion in the number density, leading to the so-called virial expansion:

$$PV = N k_B T \left(1 + B_2(T)\frac{N}{V} + B_3(T)\left(\frac{N}{V}\right)^2 + \ldots \right). \tag{4.60}$$

where $B_2(T)$ is a temperature-dependent coefficient, called the second virial coefficient. Although for high densities the third, fourth, etc, virial coefficients become important, for low densities the second virial coefficient suffices to accurately predict the system's pressure. This is exactly the range where Eq. (4.59) is valid. So we find the following equation for the second virial coefficient:

$$B_2(T) = -2\pi \int_0^\infty \mathrm{d}r\, r^2 \left(\mathrm{e}^{-\beta\varphi(r)} - 1 \right) \tag{4.61}$$

The above equation is important because it allows us to calculate the pressure of a fluid knowing only the pair interaction $\varphi(r)$ between its constituent molecules. In the next subsection we will apply this to a hard sphere fluid.

The hard sphere fluid

In many theories of liquids the hard sphere fluid is used as a reference system, to which interparticle attractions are added as a perturbation. It is therefore useful to study the radial distribution function, second virial coefficient and pressure of a hard sphere fluid.

The pair interaction in a hard sphere fluid is given by

$$\varphi(r) = \begin{cases} \infty & \text{for } r \le \sigma \\ 0 & \text{for } r > \sigma \end{cases} \tag{4.62}$$

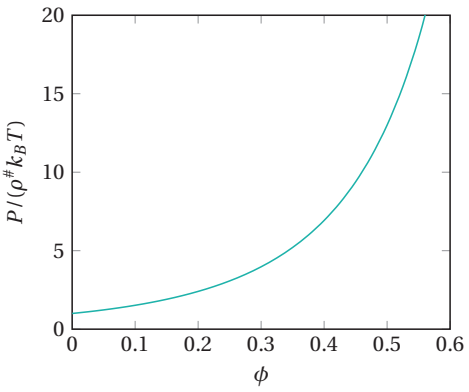

Figure 4.6: The Carnahan-Starling equation for the pressure of a hard sphere fluid as a function of solid volume fraction ϕ. Note that the prediction for very high densities $\phi > 0.5$ is incorrect since the pressure in a real hard sphere fluid diverges at $\phi_{max} \approx 0.64$ for random close packing.

At very low densities the radial distribution function and second virial coefficient are therefore given by

$$g(r) \approx \begin{cases} 0 & \text{for } r \leq \sigma \\ 1 & \text{for } r > \sigma \end{cases} \tag{4.63}$$

$$B_2 = -2\pi \int_0^\infty \mathrm{d}r r^2 \left(\mathrm{e}^{-\beta\varphi(r)} - 1\right) = 2\pi \int_0^\sigma \mathrm{d}r r^2 = \frac{2}{3}\pi\sigma^3. \tag{4.64}$$

According to Eq. (4.60), and using $\phi = \frac{1}{6}\pi\rho^{\#}\sigma^3$ for the volume fraction of spheres, the pressure of a hard sphere fluid can be expressed as:

$$P = \rho^{\#} k_B T \left(1 + 4\phi\right). \tag{4.65}$$

The above expressions are valid for not-too-high densities. At higher densities the probability to find another hard sphere in (near-)contact with a given hard sphere is higher than 1, and the pressure is higher than predicted by the second virial coefficient alone. Using a computer one has calculated that the pressure for more general densities is given by:

$$\frac{P}{\rho^{\#} k_B T} = 1 + 4\phi + 10\phi^2 + 18.365\phi^3 + 28.24\phi^4 + 39.5\phi^5 + 56.6\phi^6 + \ldots \tag{4.66}$$

This is approximately

$$\frac{P}{\rho^{\#} k_B T} = 1 + 4\phi + 10\phi^2 + 18\phi^3 + 28\phi^4 + 40\phi^5 + 54\phi^6 + \ldots \tag{4.67}$$

Extrapolating and summing we find

$$\frac{P}{\rho^{\#} k_B T} = 1 + \sum_{n=1}^{\infty} (n^2 + 3n)\phi^n = \frac{1 + \phi + \phi^2 - \phi^3}{(1-\phi)^3}. \tag{4.68}$$

This is called Carnahan and Starling's equation for the pressure of a hard sphere fluid (Figure 4.6). Monte Carlo simulations of hard sphere fluids have shown that Eq. (4.68) is nearly exact at all possible volume fractions, except near random close packing densities.

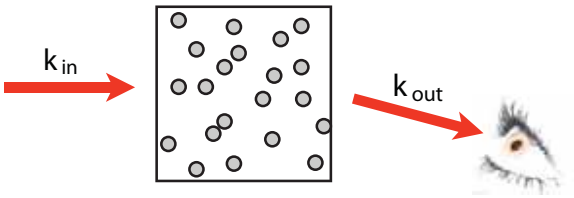

Figure 4.7: An incoming wave with wave vector $\mathbf{k}_{in}$ is scattered and analysed in the direction $\mathbf{k}_{out}$, with $|\mathbf{k}_{in}| = |\mathbf{k}_{out}|$ for elastic scattering. The scattered intensity depends on density fluctuations inside the fluid.

The static scattering function

In the precious section we have linked the compressibility of a fluid to spontaneous fluctuations in the number of particles in a large volume. More generally, density fluctuations in a fluid can be described by means of their Fourier components:

$$\rho^{\#}(\mathbf{r}) = \rho^{\#} + \frac{1}{(2\pi)^3}\int \mathrm{d}^3k\, \hat{\rho}(\mathbf{k}) \exp\{-\mathrm{i}\mathbf{k}\cdot\mathbf{r}\}, \tag{4.69}$$

$$\hat{\rho}(\mathbf{k}) = \int \mathrm{d}^3r\, \{\rho^{\#}(\mathbf{r}) - \rho^{\#}\} \exp\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}\}, \tag{4.70}$$

where $\rho^{\#} = N/V$ is the average number density of the system and $\rho^{\#}(\mathbf{r})$ the *local* number density near $\mathbf{r}$. The microscopic variable corresponding to a density Fourier component is[7]

$$\hat{\rho}(\mathbf{k}) = \int \mathrm{d}^3r \left\{ \sum_{j=1}^{N} \delta\left(\mathbf{r} - \mathbf{r}_j\right) - \rho^{\#} \right\} \exp\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}\}, \tag{4.71}$$

where $\delta(\mathbf{r}) = \delta(x)\delta(y)\delta(z)$ is the three-dimensional Dirac delta-function. This may be rewritten as

$$\begin{aligned} \hat{\rho}(\mathbf{k}) &= \sum_{j=1}^{N} \exp\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}_j\} - \rho^{\#} \int \mathrm{d}^3r \exp\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}\} \\ &= \sum_{j=1}^{N} \exp\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}_j\} - (2\pi)^3 \rho^{\#} \delta(\mathbf{k}). \end{aligned} \tag{4.72}$$

Density fluctuations in a fluid can be measured experimentally by means of scattering of light, neutrons, or X-rays (depending on the scale of interest), see Fig. 4.7. The scattered intensity also depends on details such as wave polarization and scattering strength or form factor, but generally scattering experiments measure correlation functions of Fourier components of the density. The correlation function of $\hat{\rho}(\mathbf{k})$ with its complex conjugate $\hat{\rho}^*(\mathbf{k}) = \hat{\rho}(-\mathbf{k})$, i.e. the mean square of the density fluctuation

[7] In order to avoid overly dressed symbols, we use the same symbol for the macroscopic quantity and the microscopic variable. In general a microscopic variable A^{micr} is an expression given explicitly in terms of positions and/or velocities of the particles, which after ensemble averaging yields the corresponding macroscopic quantity A, i.e. $\langle A^{micr} \rangle = A$. For example the microscopic density at $\mathbf{r}$ is given by $\rho^{micr}(\mathbf{r}) = \sum_j \delta(\mathbf{r} - \mathbf{r}_j)$, and the macroscopic density by $\rho(\mathbf{r}) = \langle \rho^{micr}(\mathbf{r}) \rangle$.

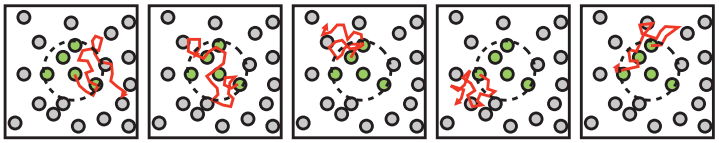

Figure 4.8: Self-diffusion: each particle, initially residing within a very small dot, will diffuse away via a different path.

with wave vector $\mathbf{k}$, is a real function of the wavevector, called the *structure factor* $S(\mathbf{k})$:

$$S(\mathbf{k}) \equiv \frac{1}{N}\left\langle \hat{\rho}(\mathbf{k})\hat{\rho}^*(\mathbf{k})\right\rangle . \tag{4.73}$$

The division by N leads to a quantity which for large enough systems is independent of system size (that is to say, the mean square density fluctuations grow linearly with system size). The structure factor gives a lot of information about the structure of a fluid. It is essentially a Fourier transform of the radial distribution function, as can be shown as follows:

$$\begin{aligned} S(\mathbf{k}) &= \frac{1}{N}\left\langle \sum_{j=1}^{N}\sum_{k=1}^{N} \exp\left\{i\mathbf{k}\cdot\left(\mathbf{r}_j-\mathbf{r}_k\right)\right\}\right\rangle - \frac{\left(\rho^{\#}\right)^2}{N}\int \mathrm{d}^3 r \int \mathrm{d}^3 r' \exp\left\{\mathrm{i}\mathbf{k}\cdot\left(\mathbf{r}-\mathbf{r}'\right)\right\} \\ &= 1+\frac{1}{N}\left\langle \sum_{j=1}^{N}\sum_{k\neq j}^{N} \exp\left\{i\mathbf{k}\cdot\left(\mathbf{r}_j-\mathbf{r}_k\right)\right\}\right\rangle - \rho^{\#}\int \mathrm{d}^3 r \exp\left\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}\right\} \\ &= 1+\rho^{\#}\int \mathrm{d}^3 r \left[g(r)-1\right]\exp\left\{\mathrm{i}\mathbf{k}\cdot\mathbf{r}\right\}. \end{aligned} \tag{4.74}$$

Comparison with Eq. (4.56) shows, perhaps surprisingly, that the compressibility of a fluid can be obtained not only by compressing the fluid and measuring the pressure, but also by performing a scattering experiment:

$$\rho^{\#} k_B T \kappa_T = \lim_{k\to 0} S(k). \tag{4.75}$$

4.6 Measuring dynamic properties

Mean-square displacement and self-diffusion

Suppose we label some particles inside a very small region (a dot) in an otherwise homogeneous fluid, at time $t = t_0$ at position $\mathbf{r}_0$, as in Fig. 4.8. When the dot, although on a macroscopic scale concentrated at $\mathbf{r}_0$, is dilute enough on a molecular scale, we may consider the concentration decay as due to the self-diffusion of the separate labeled particles. As we discussed in subsection 3.2, the conditional probability $P(\mathbf{r}, t)$ that a particle is at $\mathbf{r}$ at time t, given it was at $\mathbf{r}_0$ at time t_0, may be obtained from Fick's law:

$$\frac{\partial P(\mathbf{r}, t; \mathbf{r}_0, t_0)}{\partial t} = D_s \nabla^2 P(\mathbf{r}, t; \mathbf{r}_0, t_0), \tag{4.76}$$

together with the boundary condition $\lim_{t\to t_0} P(\mathbf{r}, 0; \mathbf{r}_0, t_0) = \delta(\mathbf{r}-\mathbf{r}_0)$. D_s is the self-diffusion coefficient, which has units of length squared over time (m^2/s). The mean

square displacement of the labeled particles can be related to the self-diffusion coefficient as follows:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}\left\langle|\mathbf{r}(t)-\mathbf{r}(t_0)|^2\right\rangle &= \int \mathrm{d}^3 r\,|\mathbf{r}(t)-\mathbf{r}(t_0)|^2\,\frac{\partial P(\mathbf{r},t;\mathbf{r}_0,t_0)}{\partial t}\\ &= D_s\int \mathrm{d}^3 r\,|\mathbf{r}(t)-\mathbf{r}(t_0)|^2\,\nabla^2 P(\mathbf{r},t;\mathbf{r}_0,t_0)\\ &= D_s\int \mathrm{d}^3 r\,P(\mathbf{r},t;\mathbf{r}_0,t_0)\nabla^2 r^2\\ &= 6D_s,\end{aligned} \tag{4.77}$$

where we have used partial integration and the fact that $P(\mathbf{r},t;\mathbf{r}_0,t_0)$ and its derivative are zero far from $\mathbf{r}_0$. For real fluid particles Fick's law only holds for large values of t.[8] Integration of Eq. (4.77) yields the Einstein equation for the self-diffusion coefficient,

$$D_s = \lim_{t\to\infty}\frac{1}{6(t-t_0)}\left\langle|\mathbf{r}(t)-\mathbf{r}(t_0)|^2\right\rangle. \tag{4.78}$$

The Einstein equation learns us that we can measure the self-diffusion coefficient of a particle by studying its mean square displacement. The pointy brackets indicate that we can average over many different time origins t_0; what is important is the *correlation time* $\tau = t - t_0$ between the current time t and the time origin t_0. To achieve a higher statistical accuracy, we should therefore use more than one time origin. How to do this is the topic of the next section.

Measuring time correlation functions

To obtain information about dynamical (transport) properties of a system, we often need to measure time correlation functions of certain variables in our particle-based simulations. These correlations functions can be in the form of the time-averaged product of a variable with the same variable some correlation time τ later (an *auto-correlation function*),

$$\langle A(\tau)A(0)\rangle_T \equiv \frac{1}{T-\tau}\int_0^{T-\tau} A(t_0+\tau)A(t_0)\mathrm{d}t_0, \tag{4.79}$$

or in the form of the time-averaged moment n of a change in a variable that occurred during a correlation time τ,

$$\left\langle[A(\tau)-A(0)]^n\right\rangle_T \equiv \frac{1}{T-\tau}\int_0^{T-\tau}[A(t_0+\tau)-A(t_0)]^n\,\mathrm{d}t_0. \tag{4.80}$$

Although such time correlation functions may be determined during post-processing, after the simulation run has finished, in practice this requires storage of a large amount

[8] At short times the fluid particles are not yet moving completely randomly. For example, they may still be trapped inside a temporary cage formed by their neighbours. Fick's law applies to time scales on which the particles are diffusing freely.

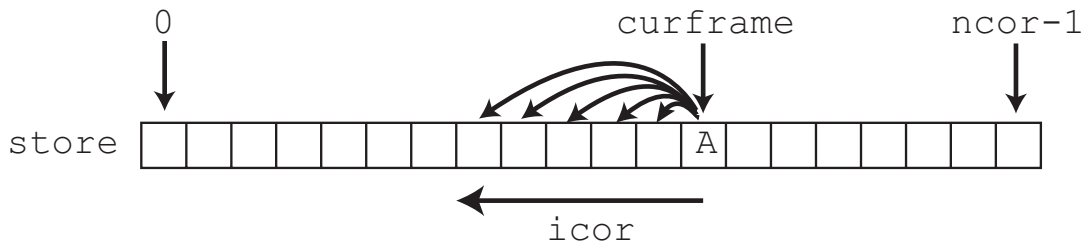

Figure 4.9: An on-the-fly calculation of time correlation functions is made possible by storing historical values of a variable in a ring buffer `store`. The last value is stored at position `curframe`, after which the autocorrelation with all possible previous values (`icor` = 0, 1, 2, etc.) is updated. `curframe` is looping from 0 to `ncor-1`, i.e. when the ring buffer is full, the oldest value of A will be overwritten by the newest.

of data from multiple time frames if the variable depends on the positions or velocities of many particles. For example, to measure the mean square displacement of particles, we would need to store in a file the positions of all particles at many different time frames. With millions of particles, this quickly becomes prohibitive.

We conclude that it is desirable to calculate time correlation functions *on the fly*, i.e. during the execution of the simulation. For the particular method we will describe here, we need to choose two timescales:

1. the smallest time resolution dτ of the time correlation function, determined by the number of integration timesteps (`num_dtau`) between taking a sample of our variable A;

2. the maximum correlation time τ_{max} that we are interested in, determined by the number of historic samples (`ncor`) that we will keep in memory at any time.

The time resolution for the time correlation function is therefore dτ =`num_dtau`$*\Delta t$ and the maximum correlation time is τ_{max} =`num_dtau*ncor`$*\Delta t$. Preferably, dτ is smaller than the fastest characteristic time scale of A and τ_{max} much larger than the longest characteristic time scale. However, often these characteristic time scales are *a priori* unknown. Often, this is actually the reason for measuring the time correlation function in the first place. If the characteristic time scales can also not be estimated from physical intuition or knowledge of similar systems, the two time scales of the correlator should be obtained by trial-and-error.

Advantage of calculating time correlations on the fly is that, as soon as the simulation time is longer than τ_{max}, we can forget about the oldest time frame, and use its storage location to store the newest time frame. We can therefore use a *ring buffer* to store data of the last `ncor` time frames, keeping track of the location `curframe` of the most recent entry (Figure 4.9).

A time correlation measurement must be initialised before the main time loop by:

```
routine initialise_correlator
frame    = 0      !current frame number
cor(:)   = 0.     !correlator bins
count(:) = 0      !normalisation bins
store(:) = 0.     !empty ring buffer storage
end routine
```

During execution of the main time loop, if the modulo of the time step with `num_dtau` is zero (t is an integer number of $d\tau$), we update the ring buffer and the correlations:

```
routine update_correlator
maxcor = ncor-1
if (frame < ncor) maxcor = frame !don't use more frames than we have
curframe = mod(frame,ncor)       !current frame number in range [0,ncor-1]
store(curframe) = A              !store current variable 'A' in ringbuffer
do icor = 0, maxcor              !correlate with 'A' in the past
  cor(icor)   = cor(icor)   + A*store(mod(curframe-icor,ncor))
  count(icor) = count(icor) + 1
enddo
frame = frame+1
end routine
```

Note that we store the current value of A in the array `store` at position `curframe`. The above routine updates the correlation between the current value of A and previous values of A for all correlation times up to the runtime or the maximum correlation time, whichever is the smallest. Also note that this routine applies to a time autocorrelation of A. If we are interested in a moment n of the change in A, we should replace just one line, namely the update of `cor(icor)`, by:

```
  cor(icor) = cor(icor) + (A-store(mod(curframe-icor,ncor)))^n
```

Finally, at the end of the simulation, we calculate the correlation function by normalising with the number of samples for each correlation time, and print the result:

```
routine finalise_correlator
do icor = 0,ncor-1
  if (count(icor)>0) print icor*num_dtau*dt, cor(icor)/count(icor)
enddo
end routine
```

Up to this point we have considered a general variable A of the system. As an explicit example, suppose we wish to calculate the mean square displacement of all particles in a simulation. If we use periodic boundary conditions this poses an additional problem, because particles that leave the central box through one face will reappear at the opposite face. For a correct measurement of the mean square displacement, we must ensure that we store the unfolded particle positions in the ring buffer. This may

be accomplished by comparing the position of a particle with the position of the same particle in the previous frame. If the particle appears to have displaced by more than half a box length in a certain cartesian direction, we should first subtract or add an integer number of box lengths to store the unfolded position. In detail:

```
routine initialise_msd
frame    = 0   !current frame number
cor(:)   = 0.  !correlator bins
count(:) = 0   !normalisation bins
store(:,:,:)= 0. !ring buffer (1st index is particle index, 2nd for x,y,z)
do i = 1, N
  store(i,1,frame) = x(i) !store current positions
  store(i,2,frame) = y(i)
  store(i,3,frame) = z(i)
enddo
frame = frame+1
end routine

routine update_msd
maxcor = ncor-1
if (frame < ncor) maxcor = frame !don't use more frames than we have
curframe  = mod(frame,ncor)      ! current frame number in range [0,ncor-1]
prevframe = mod(curframe-1,ncor) !previous frame number in range [0,ncor-1]
do i = 1, N
  deltax = x(i)-store(i,1,prevframe)
  deltay = y(i)-store(i,2,prevframe)
  deltaz = z(i)-store(i,3,prevframe)
  xunfold = store(i,1,prevframe) + deltax - Lx*nint(deltax/Lx)
  yunfold = store(i,2,prevframe) + deltay - Ly*nint(deltay/Ly)
  zunfold = store(i,3,prevframe) + deltaz - Lz*nint(deltaz/Lz)
  store(i,1,curframe) = xunfold  !store unfolded position of each particle
  store(i,2,curframe) = yunfold
  store(i,3,curframe) = zunfold
  do icor = 0, maxcor            !correlate with unfolded positions in past
    cor(icor) = cor(icor) + (xunfold-store(i,1,mod(curframe-icor,ncor)))^2
                          + (yunfold-store(i,2,mod(curframe-icor,ncor)))^2
                          + (zunfold-store(i,3,mod(curframe-icor,ncor)))^2
    count(icor) = count(icor) + 1
  enddo
enddo
frame = frame+1
end routine
```

Finalising the mean square displacement measurement can be done by `finalise_correlator`.

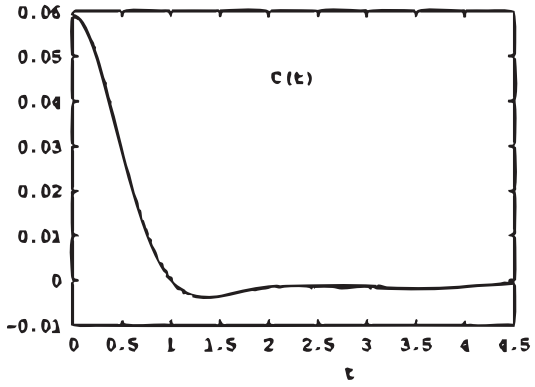

Figure 4.10: Typical velocity autocorrelation function in a liquid. In this figure the relatively fast initial decay is clearly visible, whereas the slow decay at larger times is not. Nevertheless, the slow decay contributes considerably to the self-diffusion coefficient, Eq. (4.82).

Velocity autocorrelation function and self-diffusion

Obtaining the self-diffusion coefficient from the mean square displacement is slightly complicated by the use of periodic boundary conditions. Fortunately there is also another way to obtain the self-diffusion coefficient which is not complicated by the use of periodic boundaries. To see why, let us transform the mean square displacement to an expression involving the autocorrelation of the velocity $\mathbf{v} = \dot{\mathbf{r}}$ of a labeled particle:

$$\begin{aligned}
\left\langle |\mathbf{r}(t) - \mathbf{r}(0)|^2 \right\rangle &= \int_0^t \mathrm{d}t' \int_0^t \mathrm{d}t'' \left\langle \mathbf{v}(t') \cdot \mathbf{v}(t'') \right\rangle \\
&= 2 \int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}t'' \left\langle \mathbf{v}(t') \cdot \mathbf{v}(t'') \right\rangle \\
&= 2 \int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}t'' \left\langle \mathbf{v}(t' - t'') \cdot \mathbf{v}(0) \right\rangle \\
&= 2 \int_0^t \mathrm{d}t' \int_0^{t'} \mathrm{d}\tau \left\langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \right\rangle \\
&= 2 \int_0^t \mathrm{d}\tau (t - \tau) \left\langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \right\rangle . \qquad (4.81)
\end{aligned}$$

The last step follows after partial integration. Comparing with the Einstein equation (4.78), taking the limit for $t \to \infty$, we finally find

$$D_s = \frac{1}{3} \int_0^\infty \mathrm{d}\tau \left\langle \mathbf{v}(\tau) \cdot \mathbf{v}(0) \right\rangle . \qquad (4.82)$$

This is the Green-Kubo relation for the self-diffusion coefficient.

In Fig. 4.10 a typical velocity autocorrelation is shown. After a short time the autocorrelation goes through zero; here the particle collides with some other particle in front of it and it reverses it's velocity. For large values of τ the velocity autocorrelation scales as $\tau^{-3/2}$, which is a hydrodynamic effect. This very slow decay is often difficult to detect.

Stress autocorrelation function and viscosity

In the previous subsection we have seen that a transport property (diffusion) may be measured through the integral of a time correlation function (the velocity autocorrelation) measured in equilibrium. Similarly, it may be shown (we will not do this here explicitly) that the dynamic viscosity of a molecular system may also be obtained from the integral of the time correlation function of a quantity measured in equilibrium. In this case the quantity of interest is an off-diagonal, say xy-component, of the microscopic stress tensor. In formula:

$$\mu = \frac{V}{k_B T} \int_0^\infty \mathrm{d}\tau \left\langle S_{xy}^{micr}(\tau) S_{xy}^{micr}(0) \right\rangle, \tag{4.83}$$

where V is the volume of the system, $k_B T$ the thermal energy, and S_{xy}^{micr} the xy-component of the microscopic stress tensor, given by:

$$S_{xy}^{micr} = -\frac{1}{V} \sum_{i=1}^{N} \left[m_i v_{i,x} v_{i,y} + x_i F_{i,y} \right]. \tag{4.84}$$

In the above expression m_i is the mass and $\mathbf{v}_i$ is the velocity of particle i, and $\mathbf{F}_i$ is the force on particle i due to interactions with other particles.[9] If the interactions between the particles are pairwise additive, we can also write this as:

$$S_{xy}^{micr} = -\frac{1}{V} \sum_{i=1}^{N} \left[m_i v_{i,x} v_{i,y} + \left(x_i - x_j \right) F_{ij,y} \right], \tag{4.85}$$

where $F_{ij,y}$ is the y-component of the force on i due to its interaction with j.

Because the stress tensor is a global property of the system instead of a particle property, in practice much longer simulation times are required to obtain accurate measurements of the viscosity than of the diffusion coefficient. For the velocity autocorrelation we can average over N independent measurements, whereas for the stress autocorrelation we can only average over 3 independent measurements, namely the autocorrelations of xy, xz, and yz components of the stress tensor.

4.7 Limitations of Molecular Dynamics simulations

Molecular dynamics simulations can be very accurate if accurate force fields are used for the interactions between the atoms. Still, there is a major limitation to molecular dynamics simulations: the integration time steps are typically of the order of a few femtoseconds, allowing for simulation times of up to 100 nanoseconds for system sizes of up to tens of nanometers. Even when massively parallel computers are used, employing tens of thousands of processors to a single problem, the largest system size that can be studied is of the order of 100 nanometers cubed.

[9] Note that this expression is reminiscent of the virial expression for the pressure, Eq. (4.39). In fact, the isotropic pressure P is equal to $-(S_{xx}^{micr} + S_{yy}^{micr} + S_{zz}^{micr})/3$, i.e. the average diagonal element of the stress tensor is negative the isotropic pressure.

4.8 Practicum: Molecular Dynamics simulation of liquid methane

In this practicum you will study the behaviour of a Lennard-Jones fluid, representing liquid methane, in a periodic system. You will thermostat the system and measure the particle velocity distribution, the radial distribution function, and the diffusion coefficient at different temperatures.

CHAPTER 5

THE MESOSCOPIC WORLD

5.1 Chapter objectives

Through the course of this chapter, you will accomplish the following:

- You will learn about coarse-graining and the softness of soft matter
- You will learn about the connection between dissipative forces and fluctuating forces in mesoscale systems
- You will learn about Langevin Dynamics and Brownian Dynamics simulations
- You will learn about hydrodynamic interactions between mesoscopic particles
- You will learn about simulation methods that enable hydrodynamic interactions, particularly dissipative particle dynamics and multi-particle collision dynamics
- You will learn about limitations of mesoscopic particle-based methods

5.2 Coarse-graining, soft matter systems and hydrodynamic interactions

Many systems we encounter in our daily lives consist of a continuous fluid phase with embedded particles which are much larger than single molecules. Examples include paint, blood, toothpaste, coffee, milk, and mayonnaise. The embedded particles in these systems of course ultimately also consist of atoms and molecules, but they are coherent enough for their molecules to remain grouped over a time scale longer than

our "human" time scale with which we observe or use the system. The particles can be either solid, as in colloidal suspensions of solid polymeric particles, or deformable, as in solutions with polymeric coils (paint), oil-in-water emulsions stabilised by surfactants (mayonnaise), or red blood cells in blood plasma. If the particles are less than roughly 10 micrometer in size, they are still very much influenced by thermal fluctuations. Such *mesoscopic systems* are the topic of this chapter.

It is impossible to model mesoscopic systems with atomistic detail, simply because a single particle already consists of at least millions of atoms, let alone the millons of fluid molecules surrounding each particle. So molecular dynamics simulations are out of the picture. Rather, we will treat the particles as single entities: we reduce the millions of atomic positions and velocities to a much lower number of *coarse-grained* coordinates and velocities and treat the fluid in an effective way. For example, we may describe each polymer in a polymer suspension as a string of a low number of beads, each representing the centre-of-mass of a large number of atoms, connected by springs, and treat the fluid as "blobs" of millions of fluid molecules each. This reduces the number of coordinates and velocities per polymer to only a few dozen. In the extreme case of a spherical solid particle it may be sufficient to use only 3 coordinates to describe the position of the particle and 6 coordinates to describe its velocity and angular velocity.

The equations of motion for the coarse-grained coordinates are not the same as the equations of motion for atoms in molecular dynamics simulations. We cannot simply throw away and ignore the atomistic degrees of freedom. For every configuration of coarse-grained coordinates (say a certain distance between two beads of a model polymer), there are many possible configurations of the atomistic coordinates in the particle and the fluid. First, this leads to effective interactions between coarse-grained coordinates which are generally *softer* than atomic interactions, meaning that the effective potential energy of a pair of coarse-grained particles will change much slower with their mutual distance than in the case of a pair of atoms. The resulting effective interaction energies are usually not very much higher than the thermal energy $k_B T$, meaning that these systems are easily perturbed by external forces: they are *soft matter* systems. Second, coarse-graining leads to friction forces and random forces on the coarse-grained coordinates, caused by the fluctuations of the atomistic degrees of freedom inside the particles and in the fluid. This is observed as Brownian motion of the mesoscopic particles.

The friction forces and random forces are essential ingredients of mesoscopic simulations. It is intuitively clear that the friction on a mesoscopic particle will depend on its velocity relative to the surrounding fluid. If more than one mesoscopic particle is present, the motion of one mesoscopic particle will induce a flow field in the fluid thay will alter the friction felt by another mesoscopic particle. The importance of this effect, referred to as *hydrodynamic interaction*, depends on its magnitude relative to the magnitude of other forces such as the direct particle-to-particle interaction forces. If hydrodynamic interaction can be ignored, the equations of motion are greatly simplified. This is the topic of the first part of this chapter. In the second part we will describe methods to (re-)introduce hydrodynamic interactions.

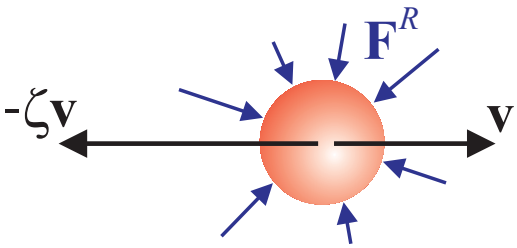

Figure 5.1: A colloidal particle moving with velocity $\mathbf{V}$ will experience a friction force $-\zeta\mathbf{V}$ opposite to its velocity and random forces $\mathbf{F}^R$ due to the continuous bombardment of solvent molecules.

5.3 Brownian motion of a single particle

Friction and random forces: the Langevin equation

Consider a spherical colloidal particle of radius a (typically between 10^{-8} and 10^{-6} meter) and mass M moving through a quiescent solvent along a path $\mathbf{R}(t)$. The colloidal particle will continuously collide with the solvent molecules. Because on average the colloid will collide more often on the front side than on the back side, it will experience a systematic force proportional with its velocity $\mathbf{V}$, and directed opposite to its velocity. The colloid will also experience a random or stochastic force $\mathbf{F}^R(t)$. These forces are summarized in Fig. 5.1 The equations of motion, referred to as the *Langevin equations*, then read

$$\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t} = \mathbf{V}, \tag{5.1}$$

$$M\frac{\mathrm{d}\mathbf{V}}{\mathrm{d}t} = -\nabla\Phi - \zeta\mathbf{V} + \mathbf{F}^R, \tag{5.2}$$

where the additional term $-\nabla\Phi$ represents a conservative force on the colloidal particle (∇ is the gradient with respect to the position of the particle). Because the colloidal particle is small, the Reynolds number associated with its motion is much lower than unity. Conversely, because the colloidal particle is much larger than the mean free path of the solvent molecules, the Knudsen number is much smaller than unity. Thus, we are allowed to obtain the friction coefficient ζ by solving the Stokes equations for fluid flow around a sphere. Assuming the colloidal particle has no-slip boundaries with the solvent,[1] it can be shown that the friction coefficient is given by the following Stokes drag formula:

$$\zeta = 6\pi\mu a, \tag{5.3}$$

where μ is the dynamic viscosity of the solvent. A full derivation is given in Appendix A.2.

[1] For very small colloidal particles in the nanometer range, the proper boundary conditions are actually quite a subtle point, which has not yet been resolved. For no-stick (full slip) boundaries, the friction coefficient is $\zeta = 4\pi\mu a$. In between these two extremes, partial-slip boundary conditions may apply at such very small scales.

The fluctuation-dissipation theorem

Assuming for the moment that no conservative forces act on the colloidal particle, $\Phi = 0$, we can divide Eq. (5.2) by M, define the friction "frequency" ξ as $\xi = \zeta/M$, and formally solve the equation, e.g. by variation of constants:

$$\mathbf{V}(t) = \mathbf{V}_0 \mathrm{e}^{-\xi t} + \frac{1}{M}\int_0^t \mathrm{d}\tau\ \mathrm{e}^{-\xi(t-\tau)}\mathbf{F}^R(t). \tag{5.4}$$

where $\mathbf{V}_0$ is the initial velocity. Note that the above solution contains a time integral over the random force $\mathbf{F}^R(t)$. We will now determine ensemble averages over all possible realizations of $\mathbf{F}^R(t)$, with the initial velocity as a condition. To this end we have to make some assumptions about the random force. In view of its chaotic character, the following assumptions seem to be appropriate for its ensemble average properties:

$$\left\langle \mathbf{F}^R(t) \right\rangle = \mathbf{0} \tag{5.5}$$

$$\left\langle \mathbf{F}^R(t)\cdot\mathbf{F}^R(t') \right\rangle_{\mathbf{V}_0} = C_{\mathbf{V}_0}\delta(t-t') \tag{5.6}$$

where $C_{\mathbf{V}_0}$ may depend on the initial velocity. The above equations say that the random force is zero on average (the average force is already included in the friction) and has no memory (a white noise process). Using Eqs. (5.4) - (5.6), we find

$$\begin{aligned}\langle \mathbf{V}(t)\rangle_{\mathbf{V}_0} &= \mathbf{V}_0\mathrm{e}^{-\xi t} + \frac{1}{M}\int_0^t \mathrm{d}\tau\ \mathrm{e}^{-\xi(t-\tau)}\left\langle \mathbf{F}^R(\tau)\right\rangle_{\mathbf{V}_0} \\ &= \mathbf{V}_0\mathrm{e}^{-\xi t} \end{aligned} \tag{5.7}$$

$$\begin{aligned}\langle \mathbf{V}(t)\cdot\mathbf{V}(t)\rangle_{\mathbf{V}_0} &= V_0^2\mathrm{e}^{-2\xi t} + \frac{2}{M}\int_0^t \mathrm{d}\tau\ \mathrm{e}^{-\xi(2t-\tau)}\mathbf{V}_0\cdot\left\langle \mathbf{F}^R(\tau)\right\rangle_{\mathbf{V}_0} \\ &\quad + \frac{1}{M^2}\int_0^t \mathrm{d}\tau'\int_0^t \mathrm{d}\tau\ \mathrm{e}^{-\xi(2t-\tau-\tau')}\left\langle \mathbf{F}^R(\tau)\cdot\mathbf{F}^R(\tau')\right\rangle_{\mathbf{V}_0} \\ &= V_0^2\mathrm{e}^{-2\xi t} + \frac{C_{\mathbf{V}_0}}{2M^2\xi}\left(1-\mathrm{e}^{-2\xi t}\right). \end{aligned} \tag{5.8}$$

The colloid is in thermal equilibrium with the solvent. According to the equipartition theorem [45], for large t, Eq. (5.8) should be equal to $3k_BT/M$, from which it follows that

$$\left\langle \mathbf{F}^R(t)\cdot\mathbf{F}^R(t')\right\rangle = 6k_BT\xi M\delta(t-t') = 6k_BT\zeta\delta(t-t'). \tag{5.9}$$

This is one manifestation of the fluctuation-dissipation theorem, which states that the dissipative force experienced by a particle dragged through a fluid is actually intimately connected to the magnitude and time correlation of the fluctuating random force.

The Einstein and Stokes-Einstein equations

Integrating Eq. (5.4) we get

$$\mathbf{R}(t) = \mathbf{R}_0 + \frac{\mathbf{V}_0}{\xi}\left(1-\mathrm{e}^{-\xi t}\right) + \frac{1}{M}\int_0^t \mathrm{d}\tau\int_0^\tau \mathrm{d}\tau'\ \mathrm{e}^{-\xi(\tau-\tau')}\mathbf{F}^R(\tau'), \tag{5.10}$$

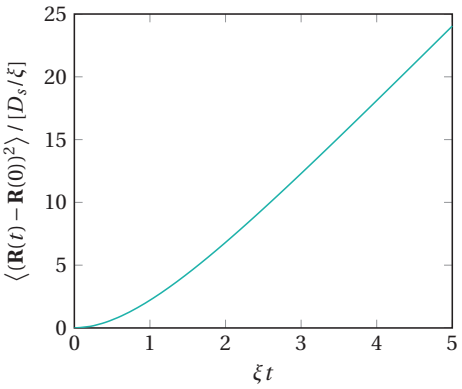

Figure 5.2: The mean square displacement of a colloidal particle according to the Langevin equation (5.11) as a function of time. Here we have used the ensemble average $\langle V_0^2 \rangle = 3k_BT/M$ for the initial velocity. The horizontal time axis is scaled by the friction frequency $\xi = \zeta/M$, and the vertical m.s.d. axis is scaled by D_s/ξ, where D_s is the self-diffusion coefficient according to the Einstein equation (5.14).

from which we calculate the mean square displacement

$$\left\langle (\mathbf{R}(t)-\mathbf{R}_0)^2 \right\rangle_{\mathbf{V}_0} = \frac{V_0^2}{\xi^2}\left(1-\mathrm{e}^{-\xi t}\right)^2 + \frac{3k_BT}{M\xi^2}\left(2\xi t - 3 + 4\mathrm{e}^{-\xi t} - \mathrm{e}^{-2\xi t}\right). \tag{5.11}$$

We can study two limits of this equation. For very small t the mean square displacement is

$$\lim_{t\downarrow 0}\left\langle (\mathbf{R}(t)-\mathbf{R}_0)^2 \right\rangle = V_0^2 t^2, \tag{5.12}$$

which is the ballistic regime when the particle has not yet significantly changed its velocity from $\mathbf{V}_0$. Conversely, for very large t the mean square displacement becomes

$$\lim_{t\to\infty}\left\langle (\mathbf{R}(t)-\mathbf{R}_0)^2 \right\rangle = \frac{6k_BT}{M\xi}t, \tag{5.13}$$

from which we get the Einstein equation for the self-diffusion coefficient[2]

$$D_s = \frac{k_BT}{\zeta}. \tag{5.14}$$

Note that we could have obtained the same result directly from Eq. (5.4) by determining the time integral of the velocity autocorrelation, and applying equipartition for the mean square velocity. Try this yourself!

When we combine the Einstein equation (5.14) with the Stokes drag equation (5.3), we find the famous *Stokes-Einstein equation*, valid for a single colloidal sphere:

$$D_s = \frac{k_BT}{6\pi\mu a}. \tag{5.15}$$

The Stokes-Einstein equation tells us, perhaps surprisingly, that the self-diffusion coefficient D_s of a colloidal particle is independent of its mass M, but only depends on its size a.

[2]The Einstein equation does not only apply to a single spherical colloidal particles, but also to collections of colloidal particles or colloidal particles of a non-spherical shape. The diffusion *tensor* $\mathbf{D}$ is then related to the friction tensor Ξ as $\mathbf{D} = k_BT\Xi^{-1}$. This is beyond the scope of these lectures.

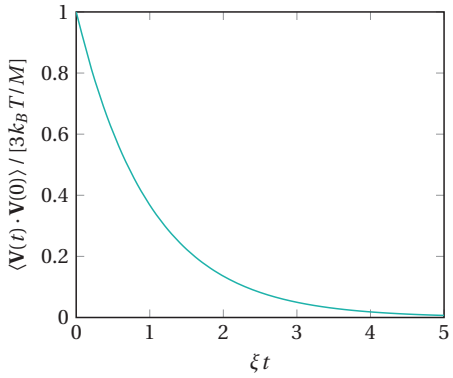

Figure 5.3: The velocity auto correlation function of a colloidal particle according to the Langevin equation (5.11) as a function of time. Time axis is scaled by the friction frequency $\xi = \zeta/M$, and the vertical velocity correlation axis is scaled by the expectation $\langle V_0^2 \rangle = 3k_BT/M$. Note that the colloid loses memory of its initial velocity after a few $1/\xi$.

Overdamped systems: from Langevin equation to Brownian dynamics

From Eq. (5.7) we see that the colloid loses its memory of its initial velocity after a time $\tau \approx 1/\xi$ (Figure 5.3). Using equipartition, its initial velocity may be put equal to $\sqrt{3k_BT/M}$. The distance λ it travels, divided by its radius then is

$$\frac{\lambda}{a} = \frac{\sqrt{3k_BT/M}}{a\xi} = \sqrt{\frac{\rho k_B T}{9\pi\mu^2 a}}, \tag{5.16}$$

where ρ is the mass density of the colloid.

When the particles are very small, have a very high density, and/or the fluid itself has a very low viscosity (e.g. the fluid is a gas), will λ/a be significant (0.1 or larger). In that case it is important to keep track of both the particle position and its velocity.

However, in many practical situations the value of λ/a is very small. For example, $\lambda/a \approx 10^{-3}$ for a 10 nanometre sized colloid and $\lambda/a \approx 10^{-4}$ for a micrometre sized colloid in water at room temperature. We see that the particles have hardly moved at the time possible velocity gradients have relaxed to equilibrium: they are overdamped. When we are interested in timescales on which particle configurations change, we may restrict our attention to the space coordinates, and average over the velocities.

It can be shown [15] (though we will not do it here) that the explicit equations of motion for the particles which only include particle coordinates are

$$\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}t} = -\frac{1}{\zeta}\nabla\Phi + \nabla D_s + \mathbf{f}^R \tag{5.17}$$

$$\langle \mathbf{f}^R(t) \rangle = \mathbf{0} \tag{5.18}$$

$$\langle \mathbf{f}^R(t)\mathbf{f}^R(t') \rangle = 2D_s\bar{\mathbf{I}}\delta(t-t'). \tag{5.19}$$

where $-\nabla\Phi$ is again a conservative force on the colloidal particle, and $\mathbf{f}^R$ are random variables with the unit of velocity. $\bar{\mathbf{I}}$ denotes the unit tensor $\mathrm{I}_{\alpha\beta} = \delta_{\alpha\beta}$, to signify that different cartesian components of the random variables $\mathbf{f}^R$ are uncorrelated. The above equations are known as the *Brownian dynamics* equations of motion. The term ∇D_s accounts for the effect of possible gradients in the self-diffusion coefficient. We emphasise again that the self-diffusion coefficient D_s and friction coefficient ζ are not independent, but linked by the Einstein equation (5.14).

5.4 Langevin and Brownian dynamics of multiple particles without hydrodynamic interactions

Neglect of hydrodynamic interactions

In most applications we are dealing with $N > 1$ particles embedded in a fluid. Compared to a single particle in a fluid, this leads to two types of additional forces:

1. direct interaction forces, usually caused by nearby other particles, and
2. hydrodynamic forces on a particle transmitted by fluid flow fields caused by the motion of other particles.

The latter, fluid-mediated, forces are referred to as hydrodynamic interactions. Because hydrodynamic fields decay very slowly with distance, as slow as a/r (see Appendix A.3), hydrodynamic interactions may be felt between particles which are many particle diameters apart.

Simulations of Brownian particles are greatly simplified if we can neglect hydrodynamic interactions. It is therefore important to understand *when* hydrodynamic interactions may be neglected.

First, hydrodynamic interactions may be neglected when the direct interaction forces between particles dominate the hydrodynamic forces. This is usually the case at high solid volume fractions ϕ of particles, where

$$\phi = \frac{N\frac{4}{3}\pi a^3}{V} = \frac{4}{3}\pi\rho^{\#}a^3 \tag{5.20}$$

is based on the hard core radius a of the particle. How high ϕ needs to be, depends on the range of the direct interaction. For example, when the colloidal particles are charged, or if their surfaces are dotted with (non-collapsed) polymer coils, particles start to feel each other long before their hard(er) cores touch. In such a case we may neglect hydrodynamic interactions as soon as the virtual spheres associated with the range of direct interaction significantly overlap. This may for example happen already for $\phi > 0.1$ when the range of direct interaction is 2-3 times the particle diameter. On the other hand, when the colloidal particles act (almost) like hard spheres, their ranges of direct interaction never significantly overlap. Indeed, for hard sphere solutions hydrodynamic forces may remain dominant for all volume fractions up to the point of maximum packing ($\phi \approx 0.64$). At the higher volume fractions these hydrodynamic interactions are mainly in the form of so-called lubrication forces. Lubrication forces are a type of hydrodynamic interaction associated with the squeezing of a fluid in and out of a region between two particles when two particle surfaces come close together or move apart.

Second, hydrodynamic interactions may be neglected at very low solid volume fractions. To see how low, we will estimate the effect of hydrodynamic flow fields on

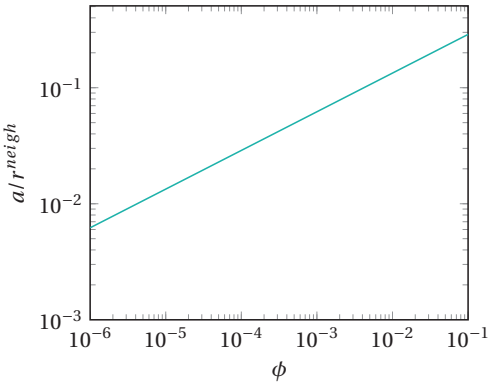

Figure 5.4: Log-log plot of the radius a of a particle divided by the typical distance r^{neigh} between neighbours in a homogeneous dispersion of particles as a function of solid volume fraction ϕ. This is a measure for the influence of hydrodynamic interactions at low ϕ.

neighbouring particles. If the particles remain dispersed through the fluid, the typical distance r^{neigh} between neighbouring particles is of the order of $r^{neigh} = (\rho^{\#})^{-1/3}$. This leads to the following estimate for the ratio a/r^{neigh}:

$$\frac{a}{r^{neigh}} \approx \left(\frac{3\phi}{4\pi}\right)^{1/3} \tag{5.21}$$

Figure 5.4 shows that if $\phi = 10^{-3}$ (or 10^{-4}), the flow field induced by a particle has decayed to 6% (or 3%) of its original value by the time it reaches the first neighbouring particle. Hydrodynamic interactions are always present, but depending on the quantity of interest and the required accuracy, hydrodynamic interactions are typically neglected for $\phi < 10^{-4}$. In this case the dynamics of a Brownian particle is similar to the dynamics of a single, isolated Brownian particle.

In summary, hydrodynamic interactions may be neglected in two extreme situations, either when the particles are so close that they significantly overlap in their direct interaction range, or when the particles are so far apart that they effectively don't feel each other anymore.

Numerical implementation of Langevin dynamics

The Langevin equations of motion for multiple particles (without hydrodynamic interactions) are

$$\frac{\mathrm{d}\mathbf{R}_i}{\mathrm{d}t} = \mathbf{V}_i, \tag{5.22}$$

$$M_i\frac{\mathrm{d}\mathbf{V}_i}{\mathrm{d}t} = \mathbf{F}_i - \zeta_i\mathbf{V}_i + \mathbf{F}_i^R, \tag{5.23}$$

$$\left\langle \mathbf{F}_i^R(t)\right\rangle = \mathbf{0} \tag{5.24}$$

$$\left\langle \mathbf{F}_i^R(t)\mathbf{F}_j^R(t')\right\rangle = 2k_BT\zeta_i\delta_{ij}\bar{\mathbf{I}}\delta(t-t'). \tag{5.25}$$

In the second line, $\mathbf{F}_i$ is the force on particle i due to direct interactions with other particles and external fields. The last line indicates that the random force on a particle

is uncorrelated with the random force on another particle (δ_{ij}), that the random force in one cartesian direction is uncorrelated with the random force in another cartesian direction ($\bar{\mathbf{I}}$), and that the random force at a time t is uncorrelated with the random force at another time t' ($\delta(t-t')$) (white noise without memory).

The random forces need some attention because in a simulation we use finite time steps Δt to integrate the equations of motion. An often used approach is to approximate the delta-correlated random force by a random force which is constant during an entire time step, but uncorrelated with the random force in another time step.[3] In that case a leap-frog or velocity Verlet algorithm (for details see section 2.7) can be used to update the positions and velocities of the particles, but with the force $\mathbf{F}_i$ on i replaced by the total force $\mathbf{F}_i^{tot}$ which includes friction and random force,

$$\mathbf{F}_i(t) \mapsto \mathbf{F}_i^{tot}(t) = \mathbf{F}_i(t) - \zeta_i \mathbf{V}_i(t) + \mathbf{F}_i^R(t), \tag{5.26}$$

where each cartesian component of $\mathbf{F}_i^R(t)$, i.e. $F_{i,x}^R$, $F_{i,y}^R$ and $F_{i,z}^R$, is sampled from a Gaussian distribution[4] with zero average and standard deviation

$$\sigma_{F_{i,\alpha}^R} = \left\langle \left(F_{i,\alpha}^R\right)^2 \right\rangle^{1/2} = \sqrt{\frac{2k_B T \zeta_i}{\Delta t}}. \tag{5.27}$$

Note that the standard deviation of the random force increases with decreasing time step Δt as $\Delta t^{-1/2}$.

Numerical implementation of Brownian dynamics

As already noted in section 5.3, using the Langevin equation to solve the motion of Brownian particles with a relatively high amount of friction ζ makes no sense because the velocities relax to equilbrium before the particle has moved only a tiny fraction of its diameter. Moreover, with increasing friction, the discretised equations of motion discussed in the previous subsection can only be used with increasingly small time steps. The fundamental limitation is that the relaxation of the velocity in one time step must be relatively small, i.e. $\xi \Delta t = \zeta \Delta t / M \ll 1$.

For particles experiencing a high friction (and for which hydrodynamic interactions can be neglected), it is more efficient to solve the velocity-averaged equations of mo-

[3] Another approach is to integrate the equations of motion over a finite time step Δt under the assumption of a constant or linearly time interpolated force $\mathbf{F}_i$. This leads, among other things, to a *correlated* random update of both the velocity and the position of a particle [].

[4] It is not absolutely necessary to sample from a Gaussian distribution. Any distribution with zero average and the same standard deviation is sufficient. This is a consequence of the central limit theorem, which states that the sum of a large number of random numbers will tend to a Gaussian distribution, irrespective of the distribution from which the original random numbers have been sampled. However, the speed with which a Gaussian distribution is approached depends, very roughly speaking, on the similarity of the original distribution to a Gaussian shape. We therefore prefer to sample from a Gaussian or near-Gaussian distribution.

tion, i.e. the Brownian equation of motion:

$$\frac{\mathrm{d}\mathbf{R}_i}{\mathrm{d}t} = \frac{\mathbf{F}_i}{\zeta_i} + \boldsymbol{\nabla} D_{s,i} + \mathbf{f}_i^R \tag{5.28}$$

$$\langle \mathbf{f}_i^R(t) \rangle = \mathbf{0} \tag{5.29}$$

$$\left\langle \mathbf{f}_i^R(t)\mathbf{f}_j^R(t') \right\rangle = 2D_{s,i}\delta_{ij}\bar{\mathbf{I}}\delta(t-t'). \tag{5.30}$$

In the first line, $\mathbf{F}_i$ is the force on particle i due to direct interactions with other particles and external fields. The last line indicates that the random variable $\mathbf{f}_i^R$ of particle i is uncorrelated with that of another particle (δ_{ij}), that it can be chosen independently for each cartesian direction ($\bar{\mathbf{I}}$), and that it has no memory ($\delta(t-t')$).

In a simulation we use a finite timestep to integrate the first-order Brownian equation of motion. Since velocities are absent, the integration is slightly different (and in fact simpler) than the leap-frog or velocity Verlet methods used to solve the second-order Langevin equations of motion. Explicitly, the integration algorithm is:

$$\mathbf{R}_i(t+\Delta t) = \mathbf{R}_i(t) + \frac{\mathbf{F}_i(t)}{\zeta_i(t)}\Delta t + \boldsymbol{\nabla}_i D_{s,i}(t)\Delta t + \Delta\mathbf{R}_i^R(t) \tag{5.31}$$

$$\left\langle \Delta R_{i,\alpha}^R \right\rangle = 0 \tag{5.32}$$

$$\sigma_{\Delta R_{i,\alpha}^R} = \left\langle \left(\Delta R_{i,\alpha}^R\right)^2 \right\rangle^{1/2} = \sqrt{2D_{s,i}(t)\Delta t}. \tag{5.33}$$

We note that the time dependence of the self-diffusion coefficient is indirectly, via its possible dependence on the particle position. Each cartesian component of the random displacement $\Delta\mathbf{R}_i^R$, i.e. $\Delta R_{i,x}^R$, $\Delta R_{i,y}^R$ and $\Delta R_{i,z}^R$, is sampled from a Gaussian distribution with zero average and standard deviation given by Eq. (5.33).

Convince yourself that for a constant diffusion coefficient ($\boldsymbol{\nabla} D_{s,i} = \mathbf{0}$) and without direct or external forces on the particles ($\mathbf{F}_i = 0$) the above algorithm leads to a mean square displacement given by $\left\langle (\mathbf{R}_i(t) - \mathbf{R}_i(0))^2 \right\rangle = 6D_{s,i}t$. Of course, when particles interact with each other, the mean square displacement may be slowed down due to temporary caging effects, i.e. in highly crowded systems the measured long time diffusions coefficient D_i may be significantly lower than the $D_{s,i}$ which is the diffusion coefficient of a single free particle.

5.5 Mesoscale methods with hydrodynamic interactions

Direct calculation of hydrodynamic interaction forces from Stokes equations: Stokesian dynamics

In many (if not most) realistic situations the hydrodynamic interactions between colloidal particles cannot be neglected. For an unbounded system consisting of slowly moving spheres in a fluid, it is possible to calculate the hydrodynamic force $\mathbf{F}_i^h$ on

sphere i due to the motion of all spheres. Details of this calculation are given in Appendix A.3. The result is:

$$\mathbf{F}_i^h = -\sum_{j=0}^{N} \bar{\zeta}_{ij} \cdot \mathbf{V}_j, \tag{5.34}$$

where to lowest order the friction tensor $\bar{\zeta}_{ij}$ is given by

$$\bar{\zeta}_{ii} = 6\pi\mu a \bar{\mathbf{I}}, \tag{5.35}$$

$$\bar{\zeta}_{ij} = -6\pi\mu a \frac{3a}{4R_{ij}} \left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{ij}\hat{\mathbf{R}}_{ij}\right) \qquad (i \neq j). \tag{5.36}$$

Here $R_{ij} = \left|\mathbf{R}_i - \mathbf{R}_j\right|$ is the distance between particle j and i, and $\hat{\mathbf{R}}_{ij} = \left(\mathbf{R}_i - \mathbf{R}_j\right)/R_{ij}$ the unit vector pointing from j to i.

It is clear from the above equations that including hydrodynamic interactions in a simulation of colloidal particles is a computationally expensive operation: all particles feel all other particles. Solving the equations of motion of all N particles on a level equivalent to the Brownian dynamics algorithm (i.e. allowing for high frictions and large time steps) requires the inversion of a $3N \times 3N$ matrix, the computational costs of which scale as N^3. This is the basis of *Stokesian dynamics* simulations [14].

The advantage of Stokesian dynamics simulations is that higher order terms of the hydrodynamic interactions can be incorporated without much additional costs, allowing for accurate results.

Unfortunately, given the high computational costs, it may come as no surprise that Stokesian dynamics simulations are typically limited to a few hundred colloidal particles. Various tricks have been applied to speed up the calculations, collectively known as *accelerated Stokesian dynamics* [57], including spliting the hydrodynamic interactions in far field and near field terms, handling the far field terms more efficiently in "wave vector space" (this is akin to the so-called Ewald summation technique for charge interactions). These tricks are rather technical and not easy to implement.

The largest drawback of Stokesian dynamics is that the expressions used for the hydrodynamic interactions apply to spheres in an unbounded medium. Non-spherical shapes may be generated by combining several spheres, but it is generally very difficult to study the behaviour of colloidal suspensions confined by walls or other boundaries.

Using particle-based mesoscale methods for hydrodynamic fluids

The high computational costs of including hydrodynamic interactions through direct calculations based on the (Navier-)Stokes equations have inspired researchers to embed colloidal particles in relatively cheap particle-based mesoscale methods for hydrodynamic fluids.

The idea is illustrated in Fig. 5.5. By excluding fluid from regions occupied by colloidal particles, and applying appropriate boundary conditions, movement of a colloidal particle will set in motion a flow in the fluid which will affect other colloidal particles, just as in a real colloidal dispersion.

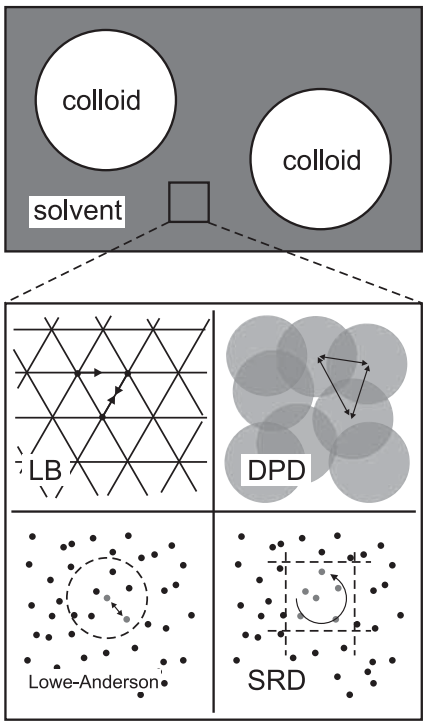

Figure 5.5: Schematic picture depicting how a colloidal dispersion can be coarse-grained by replacing the fluid with a particle-based hydrodynamics solver such as lattice Boltzmann (LB), dissipative particle dynamics (DPD), the Lowe-Anderson thermostat, or stochastic rotation dynamics (SRD), which is a type of multi-particle collision dynamics (MPCD) [51]. Each method introduces an effective coarse-graining length scale that is chosen to be smaller than those of the mesoscopic colloids but much larger than the natural length scales of a microscopic fluid. By obeying local momentum conservation, these methods reproduce Navier-Stokes hydrodynamics on larger length scales. For LB, thermal fluctuations must be added in separately, but these emerge naturally for the other three methods.

A large number of particle-based mesoscale methods for hydrodynamic fluids exist. The best known are:

- DSMC: direct simulation Monte Carlo [10],
- LA: Lowe-Andersen method [40, 41],
- LGA and LBM: lattice gas automata and lattice Boltzmann methods [58],
- DPD: dissipative particle dynamics [23, 25, 31],
- MPCD: multi-particle collision dynamics [24, 29, 42, 51],

All these methods follow the idea of coarse-graining the countless molecules in a fluid to a much lower number of particles. The coarse-grained fluid retains its viscous properties by performing efficient collisions between the fluid particles. The collisions are constructed such that they obey local conservation of mass, momentum and (in most cases) energy. These local conservation laws are sufficient to generate a correct Navier-Stokes hydrodynamics in the continuum limit.

In direct simulation Monte Carlo (DSMC) simulations, collisions between neighbouring particles are executed probabilistically, based on the relative velocities and particle sizes using kinetic theory of gases [10]. DSMC is particularly useful for rarefied gas flows, where the Knudsen number is of the order of 1 or larger.

In the Lowe-Andersen (LA) method, with a certain frequency random neighbouring pairs of particles within a certain range are picked to collide with each other [40, 41].

The collision is executed by resampling the relative velocity (the component along the line connecting the pair of particles) from a Maxwell-Boltzmann distribution.

In lattice gas automata (LGA), particles live on a lattice and can attain a low number of discrete values for the velocity [58]. Collisions are executed with probabilities chosen such that the relaxation of hydrodynamic velocity fluctuations is isotropic on sufficiently large length scales. Lattice Boltzmann methods (LBM) are based on the same *idea* of particles colliding on a lattice, but are propagating the probability density distribution according to the (linearised and pre-averaged) Boltzmann equation [58]. Thermal fluctuations, which are crucial for mesoscopic systems, are absent in the original Lattice Boltzmann method, but can be added through a fluctuating stress tensor [1,20].

In these lectures we will focus on the two last common hydrodynamic mesoscale methods, namely dissipative particle dynamics and multi-particle collision dynamics.

5.6 Dissipative particle dynamics

Two innovations

Dissipative particle dynamics (DPD) [23,25,31] is a popular particle-based method that includes hydrodynamics and thermal fluctuations. It can be viewed as an extension of standard (Newtonian) molecular dynamics techniques, but with two important innovations:

1. soft potentials that allow large time steps and rapid equilibration,
2. a Galilean-invariant[5] thermostat that locally conserves momentum and therefore generates correct Navier-Stokes hydrodynamics.

The statistical mechanical origins of innovation (1) of DPD, the use of soft potentials out of equilibrium, are still under debate, but are often explained as resulting from viewing the DPD particles as "clumps" of the underlying fluid.

Innovation (2), on the other hand, can be put on firmer statistical mechanical footing, and can be usefully employed to study the dynamics of complex systems with other types of interparticle interactions. The main advantage of the DPD thermostat is that, by preserving momentum conservation, the hydrodynamic interactions that intrinsically arise from microcanonical MD are preserved for calculations in the canonical ensemble. Other thermostats (such as the ones treated in section 4.4) typically screen the hydrodynamic interactions beyond a certain length scale because they introduce an effective friction with a hypothetical stagnant background.[6] For weak damping this may not be a problem, but for strong damping it could be.

[5] *Galilean invariant* means that the behaviour of the system is exactly the same in another inertial frame of reference. In other words, when the velocities of all particles are shifted by a constant amount $\mathbf{V}$, the updates of particle velocities are exactly the same as in the original system.

[6] The molecular dynamics thermostats introduced in section 4.4 rescale absolute velocities, meaning that macroscopic flow velocities are dampened to zero. These thermostats are *not* Galilean invariant.

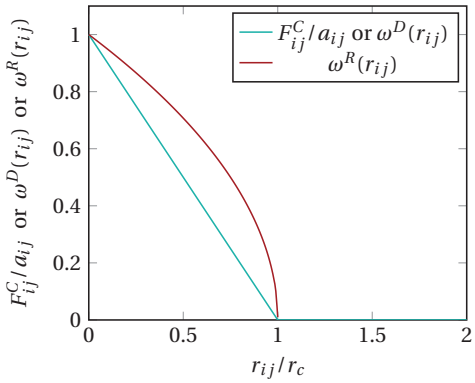

Figure 5.6: In dissipative particle dynamics, the conservative force F^C_{ij} and dissipative force F^D_{ij} decay linearly from a maximum value for pair distance $r_{ij} = 0$ to zero for pair distance $r_{ij} = r_c$ (green line). The strength of the random force decays as the square root of this (red line).

Simulating a pure DPD fluid

In DPD, the force on each particle i results from a sum of pair-wise forces:

$$\mathbf{F}_i = \sum_{j \neq i} \mathbf{F}^C_{ij} + \mathbf{F}^D_{ij} + \mathbf{F}^R_{ij}, \tag{5.37}$$

with

$$\mathbf{F}^C_{ij} = \begin{cases} a_{ij}\left(1 - \frac{r_{ij}}{r_c}\right)\hat{\mathbf{r}}_{ij} & \text{for } r_{ij} \leq r_c \\ 0 & \text{for } r_{ij} > r_c \end{cases} \tag{5.38}$$

$$\mathbf{F}^D_{ij} = -\gamma \omega^D(r_{ij})\left[(\mathbf{v}_i - \mathbf{v}_j)\cdot\hat{\mathbf{r}}_{ij}\right]\hat{\mathbf{r}}_{ij} \tag{5.39}$$

$$\mathbf{F}^R_{ij} = \sqrt{\frac{2\gamma k_B T \omega^D(r_{ij})}{\Delta t}}\xi_{ij}\hat{\mathbf{r}}_{ij} \tag{5.40}$$

Here $r_{ij} = \left|\mathbf{r}_i - \mathbf{r}_j\right|$ is the distance between particle i and j and $\hat{\mathbf{r}}_{ij} = (\mathbf{r}_i - \mathbf{r}_j)/r_{ij}$ is the unit vector pointing from particle j to particle i.

$\mathbf{F}^C_{ij}$ is a soft conservative pair force mimicking internal pressure and Van der Waals interactions. It decays linearly from an amplitude a_{ij} for fully overlapping particles ($r_{ij} = 0$) to zero at a cut-off radius r_c, see Figure 5.6. The parameter a_{ij} controls the strength of this repulsive force and solvation properties. A typical value that reproduces the properties and compressibility of water is $a_{ij} = 75 k_B T/(\rho^{\#} r_c^4)$ for relatively high number densities $\rho^{\#} > 3 r_c^{-3}$ [25]. Contrary to, e.g., the Lennard-Jones force the conservative force does not diverge for small distances.[7] This allows for relatively large integration time steps.

$\mathbf{F}^D_{ij}$ is a dissipative pair force mimicking the effects of viscosity of the fluid. Its strength is set by the friction coefficient γ and it depends on the pair distance through a weight function $\omega^D(r_{ij})$. The weight function is usually also a linearly decaying func-

[7] Note that is not forbidden to use a different interaction together with the DPD dissipative and friction forces. The linear force Eq. (5.38) is the most common in DPD literature.

tion, with the same cut-off distance r_c at the conservative force (Figure 5.6):

$$\omega^D(r_{ij}) = \begin{cases} \left(1 - \frac{r_{ij}}{r_c}\right) & \text{for } r_{ij} \le r_c \\ 0 & \text{for } r_{ij} > r_c \end{cases} \tag{5.41}$$

$\mathbf{F}^R_{ij}$ is a random pair force mimicking the kinetic energy input from the microscopic degrees of freedom that have been coarse-grained out. In Eq. (5.40) we assumed, as in Langevin dynamics, that the random force is constant during an integration time step Δt; ξ_{ij} is a symmetric random number ($\xi_{ij} = \xi_{ji}$) with zero mean and unit variance. It should come as no surprise that the magnitude and distance dependence $\omega^R(r_{ij})$ of the random pair force is linked to the magnitude and distance dependence of the dissipative force. This is another consequence of the fluctuation-dissipation theorem we encountered already in section 5.3.

The total force $\mathbf{F}_i$ on a DPD particle i is used to update its velocity and position, in exactly the same way as in standard molecular dynamics, for instance by the velocity Verlet algorithm introduced in section 2.7.

The dissipative and random forces are pair forces obeying Newton's third law: $\mathbf{F}^D_{ij} = -\mathbf{F}^D_{ji}$ and $\mathbf{F}^R_{ij} = -\mathbf{F}^R_{ji}$. This is exactly what leads to a local conservation of momentum, and hydrodynamic behaviour at larger length scales.

Embedding colloidal particles in a DPD fluid

A DPD fluid may be used to transmit hydrodynamic interactions between colloidal particles. This may be accomplished by modelling a colloidal particle either as an extra large DPD particle with altered interaction forces with the surrounding DPD fluid, or by freezing multiple DPD spheres together into a "raspberry" model.

For the first option we could choose the conservative force on a colloid c of radius a due its interaction with a DPD particle i as a shifted equivalent of the DPD fluid-fluid interaction (Figure 5.7):

$$F^C_{ci} = a_c \frac{\delta_{ci}(r_{ci})}{r_c} \hat{\mathbf{r}}_{ci}, \tag{5.42}$$

where a_c is the strength of the interaction, r_{ci} is the distance between the centres of particle i and colloid c, $\hat{\mathbf{r}}_{ci}$ the unit vector pointing from particle i to colloid c, and the (positive) overlap δ is found by subtracting r_{ci} from the colloid radius a plus cut-off range r_c:

$$\delta_{ci}(r_{ci}) = \begin{cases} a + r_c - r_{ci} & \text{for } r_{ci} \le a + r_c \\ 0 & \text{for } r_{ci} > a + r_c \end{cases} \tag{5.43}$$

Direct colloid-colloid interactions can be included similarly, by an interaction shifted over $2a$, or by including harder interactions approximating hard spheres. In any case, for the fluid to behave approximately as a continuum liquid, the radius a of the colloidal particle should be much larger than the interaction range r_c of the fluid particles.

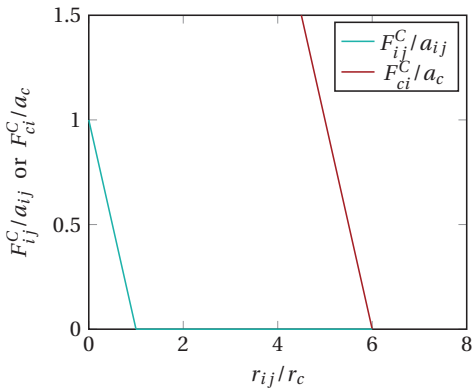

Figure 5.7: A colloidal particle may be embedded in a DPD fluid by choosing a conservative force between colloid and fluid DPD particle (red) which is shifted by the colloidal radius a compared to the fluid-fluid conservative force (green). In this example the radius and shift are $a = 5r_c$.

In practice, $a \geq 5r_c$ may already be large enough. Note that by choosing the range r_c and strength a_c of the colloid-DPD interaction, the solvability of the colloidal particle in the DPD fluid can be tuned. Also note that the forces always point in a radial direction. As a consequence, no torque can be applied to the colloidal sphere. This method therefore corresponds to slip-boundary colloidal particles.

For the second option a large number of DPD particles (typically a few hundred) is frozen into a more-or-less spherical assembly and remain so for the rest of the simulation. The assembly is allowed to translate and rotate, but the internal degrees of freedom remain fixed. Each colloidal DPD particle interacts with free DPD particles in the surrounding fluid in the same way as two free DPD particles interact, except that the force on the colloidal DPD particle is not used to directly update its position and velocity. Rather, all forces and torques relative to the centre of mass of the assembly are summed, and a rigid body update is applied using the total mass and moment of inertia tensor of the assembly. Because the free DPD particles can also apply transversal forces on the colloidal assembly, this method corresponds to stick-boundary colloidal particles.

Advantages and disadvantages of DPD

The advantages of DPD are many. The method is a simple extension of molecular dynamics. The method has been long around, so a lot of experience has been gained. The interactions are soft, allowing for relatively large (compared to molecular dynamics simulations) time steps. Coupling to colloidal particles is also easy, when treating them as extra large DPD particles.

However, there are also disadvantages. First, just as the case of molecular interactions, the repulsive interactions between the particles cause an ordering in the fluid structure, especially near walls and colloidal particles, that may be felt as oscillating conservative interactions between the colloidal particles when their surfaces are closer than a couple of r_c apart. For a molecular fluid these oscillations extend only for a few Å, much smaller than the typical colloidal radius a, whereas for a DPD fluid the oscillations extend over a distance comparable to the colloidal radius.

Second, to be in the hydrodynamic limit, colloidal particles need to be much larger than the DPD fluid particles. In the volume occupied by each colloidal particle, we typically have a few hundred DPD fluid particles. The number of DPD fluid particles quickly becomes extremely large, especially at lower colloidal volume fractions. This points to the final disadvantage: despite the gains achieved over molecular dynamics (lower number of particles, larger time step), the interactions between the DPD particles are still based on pair interactions. Even with a smart use of cell linked lists and/or neighbourlists, this means that the computational effort increases faster-than-linear with the number of DPD particles N (typically as $N \ln N$, as in molecular dynamics).

5.7 Multi-particle collision dynamics

A brief history of MPCD

In 1999, Malevanets and Kapral [42] derived a method now called stochastic rotation dynamics (SRD). In many ways it resembles the much older direct simulation Monte Carlo (DSMC) method of Bird [10]. The particles are ideal, and move in a continuous space, subject to Newton's laws of motion. At discrete timesteps, a coarse-grained collision step allows particles to exchange momentum. In SRD, space is partitioned into a rectangular grid, and at discrete time-steps the particles inside each cell exchange momentum by rotating their velocity vectors relative to the center of mass velocity of the cell. One can imagine other collision rules as well. As long as they locally conserve momentum and energy, they generate the correct Navier Stokes hydrodynamics. We may therefore view SRD as a specific implementation of the more general class of multi-particle collision dynamics (MPCD) methods.

Soon after its introduction, it was pointed out by Ihle and Kroll that for MPCD it is necessary to include a grid-shift procedure to enforce Galilean invariance [33].

An important advantage of SRD is that its simplified dynamics have allowed the analytic calculation of several transport coefficients [34, 36, 53], greatly facilitating its use.

SRD can be applied to directly simulate flow [3,38], but its stochastic nature means that a noise average must be performed to calculate flow lines, and this may may make it less efficient than pre-averaged methods like LB. Where SRD becomes more attractive is for the simulation of complex particles embedded in a solvent. Coupling can be achieved by allowing direct collisions, obeying Newton's laws of motion, between the SRD solvent and the suspended particles [51]. Such an approach is important for systems, such as colloidal suspensions, where the solvent and colloid time and length-scales need to be clearly separated. In these lectures we will focus on this latter case.

Simulating a pure SRD fluid

In SRD, the solvent is represented by a large number N (typically 10^6 or more) of particles of mass m. Here and in the following, we will call these "fluid" particles, with the caveat that, however tempting, they should not be viewed as some kind of composite particles or clusters made up of the underlying (molecular) fluid. Instead, SRD should be interpreted as a Navier-Stokes solver that includes thermal noise. The particles are merely a convenient computational device to facilitate the coarse-graining of the fluid properties.

In the first (propagation) step of the algorithm, the positions and velocities of the fluid particles are propagated for a time Δt_c (the time between collision steps) by accurately integrating Newton's equations of motion,

$$m\frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{f}_i, \tag{5.44}$$

$$\frac{\mathrm{d}\mathbf{r}_i}{\mathrm{d}t} = \mathbf{v}_i. \tag{5.45}$$

$\mathbf{r}_i$ and $\mathbf{v}_i$ are the position and velocity of fluid particle i, respectively while $\mathbf{f}_i$ is the total (external) force on particle i, which may come from an external field such as gravity, or fixed boundary conditions such as hard walls, or moving boundary conditions such as suspended colloids. The direct forces between pairs of fluid particles are, however, neglected in the propagation step. Herein lies the main advantage – the origin of the efficiency – of SRD. Instead of directly treating the interactions between the fluid particles through pair forces (as in molecular dynamics and dissipative particle dynamics), a coarse-grained collision step is performed at each time-step δt_c: First, space is partitioned into cubic cells of volume a_0^3 (Fig. 5.8), resulting in *on average* $\gamma = \rho^{\#} a_0^3$ SRD particles per cell (more on how to choose a_0 or γ later). Next, for each cell, the particle velocities relative to the center of mass velocity $\mathbf{v}_{\mathrm{cm}}$ of the cell are then rotated:

$$\mathbf{v}_{\mathrm{cm}} = \frac{\sum_{i\in\mathrm{cell}} m_i\mathbf{v}_i}{\sum_{i\in\mathrm{cell}} m_i} \tag{5.46}$$

$$\mathbf{v}_i' = \mathbf{v}_{\mathrm{cm}} + \Omega\left(\mathbf{v}_i - \mathbf{v}_{\mathrm{cm}}\right). \tag{5.47}$$

Ω is a rotation matrix which rotates velocities by a fixed angle α around a randomly oriented axis. The aim of the collision step is to transfer momentum between the fluid particles while conserving the total momentum and energy of each cell. Check for yourself that this is indeed the case. Both the collision and the streaming step conserve phase-space volume, and it has been shown that the single particle velocity distribution evolves to a Maxwell-Boltzmann distribution [42].

The rotation procedure can thus be viewed as a coarse-graining of particle collisions over space *and* time. Because mass, momentum, and energy are conserved locally, the correct hydrodynamic (Navier Stokes) equations are captured in the continuum limit, *including* the effect of thermal noise [42].

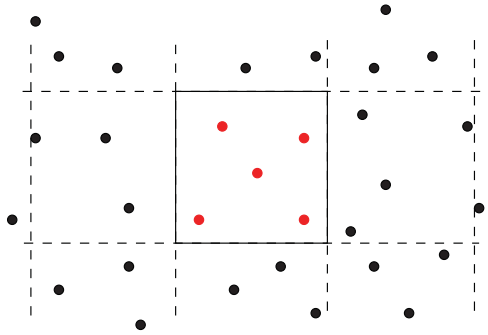

Figure 5.8: In multi-particle collision dynamics simulations, the fluid is represented by point particles. In the collision step, space is partitioned into cubic cells (lines), and all particles in a cell (e.g. all red particles) exchange momentum simultaneously.

At low temperatures or small collision times δt_c, the transport coefficients of SRD, as originally formulated [42], show anomalies caused by the fact that fluid particles in a given cell can remain in that cell and participate in several collision steps [33]. Under these circumstances the assumption of molecular chaos and Galilean invariance are incorrect. However, this anomaly can be cured by applying a random shift of the cell coordinates before the collision step [33,34].[8]

From the above description, it is clear that it is important to be able to quickly identify all particles residing in a certain cell. We already know how to do this efficiently: through a cell-linked-list (see section 2.4)! When creating the linked list we should take into account the random grid shift.

The following pseudocode generates a cell-linked-list with a random grid shift stored globally as `(shiftx,shifty,shiftz)` Here we have assumed periodic boundary conditions.[9]

```
routine generate_cell_linked_list_with_gridshift
head(:) = 0
list(:) = 0
shiftx = ran()    %a uniform random number in [0,1) (excluding 1)
shifty = ran()    %note that the same gridshift applies to all cells
shiftz = ran()    %in the current time step
do i = 1,N
  ix = int((x(i)+shiftx)*mxLx)
  iy = int((y(i)+shifty)*myLy)
```

[8] It should also be noted that the collision step in SRD does not locally conserve angular momentum. As a consequence, the stress tensor $\bar{\mathbf{S}}$ is not, in general, a symmetric function of the derivatives of the flow field (although it is still rotationally symmetric) [53]. The asymmetric part can be interpreted as a viscous stress associated with the vorticity $\nabla \times \mathbf{v}$ of the velocity field $\mathbf{v}(\mathbf{r})$. The stress tensor will therefore depend on the amount of vorticity in the flow field. However, the total force on a fluid element is determined not by the stress tensor itself, but by its divergence $\nabla \cdot \bar{\mathbf{S}}$, which is what enters the Navier Stokes equations. Taking the divergence causes the explicit vorticity dependence to drop out (the gradient of a curl is zero).

[9] If walls are present in a certain direction, we should allow for one more cell in each cartesian direction because of the random grid shift. For example, `x(i)+rx` can be in the range between 0 and $L_x + 1$. Therefore with walls in all directions, the cell index should be calculated as `icell = 1 + ix + iy*(Mx+1) + iz*(Mx+1)*(My+1)`.

```
  iz = int((z(i)+shiftz)*mzLz)
  icell = 1 + mod(ix,Mx) + mod(iy,My)*Mx + mod(iz,Mz)*Mx*My
  list(i) = head(icell)
  head(icell) = i
enddo
end routine
```

Here `mxLx = Mx/Lx` is the inverse of the cell size in the x-direction, and similarly for the other directions. In practice, in most SRD codes the lengths are expressed in units of one cell size a_0, i.e. $a_0 \equiv 1$ and masses in units of an SRD mass m, i.e. $m \equiv 1$. Because of the first choice, the multiplication by `mxLx` etc. can be avoided, making the above routine even more efficient. Because of the second choice the rotation procedure can be made more efficient, as we will see next.

The cell-linked-list must be updated each time right before the rotation procedure is executed. The rotation procedure Eq. (5.47) mimics the collisions between the SRD particles and involves choosing a randomly oriented axis independently for each cell (in 2D simulations the only choice is up or down with equal probability), and then rotating all relative velocities over an angle α around this axis. The following pseudocode performs a rotation of $\alpha = \pi/2$ around a random axis for particles that have just been sorted into a cell-linked-list by the previous routine. Here we assume that the mass of an SRD particle is unity.

```
routine collide_particles
do icell = 1,Mx*My
  i = head(icell)
  vcomx = 0.
  vcomy = 0.
  vcomz = 0.
  totm  = 0.
  do while (i.ne.0)
    vcomx = vcomx+vx(i)
    vcomy = vcomy+vy(i)
    vcomz = vcomz+vz(i)
    totm  = totm +1.0
    i = list(i)
  enddo
  if (totm > 1.0) then      %at least two particles in cell
    vcomx = vcomx/totm      %determine centre of mass velocity
    vcomy = vcomy/totm
    vcomz = vcomz/totm
    az  = 2.0*ran() - 1     %choose a random axis (ax,ay,az)
    rho = sqrt(1.0-az*az)   %of unit length
    phi = 2*pi*ran()
    ax  = rho*cos(phi)
```

```
    ay  = rho*sin(phi)
    rot = (ax*ax,    ax*ay-az, ax*az+ay;
           ax*ay+az, ay*ay,    ay*az-ax;  %rotation matrix
           ax*az-ay, ay*az+ax, az*az    )
    i = head(icell)
    do while (i.ne.0)
      vrel = (vx(i)-vcomx;  %determine relative velocity
              vy(i)-vcomy;
              vz(i)-vcomz)
      vrot = MATRIXMUL(rot,vrel)
      vx(i) = vcomx+vrot(1) %determine new velocity
      vy(i) = vcomy+vrot(2)
      vz(i) = vcomz+vrot(3)
      i = list(i)
    enddo
  endif
enddo
end routine
```

Here `MATRIXMUL` executes a matrix multiplication of the rotation matrix `rot` with the vector of relative velocities `vrel` and stores the result in a vector of rotated velocities `vrot`. For a rotation over an angle α other than $\pi/2$, the rotation matrix should be adjusted. Note that it only makes sense to apply the rotation procedure if the cell contains at least 2 SRD particles, hence the check for `totm > 1`. This is also used to prevent division by zero in the calculation of the centre of mass velocity if a cell happens to have *no* particles.

Transport properties of an SRD fluid

The simplicity of SRD collisions has facilitated the analytical calculation of many transport coefficients [34–36, 53]. These analytical expressions are particularly useful because they enable us to efficiently tune the viscosity and other properties of the fluid, without the need for trial and error simulations. In this subsection we will summarize a number of these transport coefficients, where possible giving a simple derivation of the dominant physics.

Units and the dimensionless mean-free path

As already mentioned, in most SRD simulation codes, lengths are in units of collision cell-size a_0 and masses in units of the mass m of an SRD particle. In simulations of mechanical systems only three independent scales can be set, so we have one scale left. Some authors choose the collision time interval δt_c as the third unit, but more often the thermal energy $k_B T$ is chosen as the third unit. In the examples of these lectures we will also choose $k_B T \equiv 1$.

Table 5.1: Units and simulation parameters for an SRD fluid with embedded colloidal particles. The parameters listed in the table all need to be independently fixed to determine a simulation.

Basic Units		Derived Units	
a_0	= length	$t_0 = a_0\sqrt{\frac{m}{k_BT}}$	= time
k_BT	= energy	$D_0 = \frac{a_0^2}{t_0} = a_0\sqrt{\frac{k_BT}{m}}$	= diffusion constant
m	= mass	$\nu_0 = \frac{a_0^2}{t_0} = a_0\sqrt{\frac{k_BT}{m}}$	= kinematic viscosity
		$\mu_0 = \frac{\gamma m}{t_0 a_0} = \frac{\sqrt{mk_BT}}{a_0^2}$	= viscosity

Independent fluid simulation parameters	
γ	= average number of particles per cell
δt_c	= SRD collision time interval
α	= SRD rotation angle
L	= box length

Independent colloid simulation parameters	
Δt_{MD}	= MD time step
σ_{cc}	= colloid-colloid collision diameter Eq. (5.57)
ϵ_{cc}	= colloid-colloid energy scale Eq. (5.57)
σ_{cf}	= colloid-fluid collision diameter Eq. (5.58)
ϵ_{cf}	= colloid-fluid energy scale Eq. (5.58)
N_c	= number of colloids
M_c	= colloid mass

All other units can therefore be expressed in units of a_0, m and k_BT. Time, for example, is expressed in units of $t_0 = a_0\sqrt{m/k_BT}$, the number density $n_f = \gamma/a_0^3$ and other derived units can be found in table 5.1. It is instructive to express the transport coefficients and other parameters of the SRD fluid in terms of the dimensionless mean-free path

$$\lambda = \frac{\delta t_c}{a_0}\sqrt{\frac{k_BT}{m}} = \frac{\delta t_c}{t_0}, \tag{5.48}$$

which provides a measure of the average fraction of a cell size that a fluid particle travels between collisions.

This particular choice of units helps highlight the basic physics of the coarse-graining method. The (nontrivial) question of how to map them on to the units of real physical system will be discussed later in this section.

Fluid self-diffusion coefficient

A simple back of the envelope estimate of the self-diffusion coefficient D_f of a fluid particle can be obtained from a random-walk picture. In a unit of time t_0, a particle will experience $1/\lambda$ collisions, in between which it moves an average distance λa_0. By viewing this motion as a random walk, the diffusion coefficient for a pure SRD fluid particle is therefore given by $D_f/D_0 \approx \lambda$, expressed in units of $D_0 = a_0^2/t_0 = a_0\sqrt{k_BT/m}$.

A more systematic derivation of the diffusion coefficient of a fluid particle, but still within a random collision approximation, results in the following expression [34, 55]:

$$\frac{D_f}{D_0} = \lambda\left[\frac{3}{2(1-\cos(\alpha))}\left(\frac{\gamma}{\gamma-1}\right) - \frac{1}{2}\right]. \tag{5.49}$$

The dependence on γ is weak. If, for example, we take $\alpha = \pi/2$, the value used in this paper, then $\lim_{\gamma\to\infty} D_f/D_0 = \lambda$, the same as estimated above.

While Eq. (5.49) is accurate for larger mean-free paths, where the random collision approximation is expected to be valid, it begins to show deviations from simulations for $\lambda \lesssim 0.6$ [55], when longer-time kinetic correlations begin to develop. These correlations induce interactions of a hydrodynamic nature that enhance the diffusion coefficient for a fluid particle. For example, for $\lambda = 0.1$ and $\alpha = \frac{3}{4}\pi$, the fluid self-diffusion constant D_f is about 25% larger than the value found from Eq. (5.49).

Kinematic viscosity

The spread of a velocity fluctuation $\delta\mathbf{v}(\mathbf{r})$ in a fluid can be described by the three-dimensional equivalent of Eq. (3.16):

$$\frac{\partial\delta\mathbf{v}(\mathbf{r})}{\partial t} = \nu\nabla^2\delta\mathbf{v}(\mathbf{r}) \tag{5.50}$$

where ν is the kinematic viscosity, which determines the rate at which momentum or vorticity "diffuses away". For our coice of units, the unit of kinematic viscosity is $\nu_0 = a_0^2/t_0 = a_0\sqrt{k_B T/m}$ which is the same as that for particle self diffusion i.e. $D_0 = \nu_0$.

Momentum diffusion occurs through two mechanisms:

1. By particles streaming between collision steps, leading to a "kinetic" contribution to the kinematic viscosity ν_{kin}. Since for this gas-like contribution the momentum is transported by particle motion, we expect ν_{kin} to scale like the particle self-diffusion coefficient D_f, i.e. $\nu_{kin}/\nu_0 \sim \lambda$.

2. By momentum being re-distributed among the particles of each cell during the collision step, resulting in a "collisional" contribution to the kinematic viscosity $\nu_{\rm col}$. This mimics the way momentum is transferred due to inter-particle collisions, and would be the dominant contribution in a dense fluid such as water at standard temperature and pressure. Again a simple random-walk argument explains the expected scaling in SRD: Each collision step distributes momentum among particles in a cell, making a step-size that scales like a_0. Since there are $1/\lambda$ collision steps per unit time t_0, this suggests that the collisional contribution to the kinematic viscosity should scale as $\nu_{col}/\nu_0 \sim 1/\lambda$.

Accurate analytical expressions for the kinematic viscosity $\nu = \nu_{kin} + \nu_{\rm col}$ of SRD have been derived [35, 36], and these can be rewritten in the following dimensionless form:

$$\frac{\nu_{\rm kin}}{\nu_0} = \frac{\lambda}{3} f^{\mu}_{\rm kin}(\gamma,\alpha) \tag{5.51}$$

$$\frac{\nu_{\rm col}}{\nu_0} = \frac{1}{18\lambda} f^{\nu}_{\rm col}(\gamma,\alpha). \tag{5.52}$$

where the dependence on the collisional angle α and fluid number density γ is subsumed in the following two factors:

$$f^{\mu}_{\rm kin}(\gamma,\alpha) = \frac{15\gamma}{(\gamma-1+\mathrm{e}^{-\gamma})(4-2\cos(\alpha)-2\cos(2\alpha))} - \frac{3}{2}, \tag{5.53}$$

$$f^{\mu}_{\rm col}(\gamma,\alpha) = (1-\cos(\alpha))(1-1/\gamma+\mathrm{e}^{-\gamma}/\gamma). \tag{5.54}$$

These factors only depend weakly on γ for the typical parameters used in simulations. For example, at $\alpha = \pi/2$ and $\gamma = 5$, the angle and number density we routinely use, they take the values $f^{\mu}_{\rm kin}(5,\pi/2) = 1.620$ and $f^{\mu}_{\rm col}(5,\pi/2) = 0.801$. For this choice of collision angle α, they monotonically converge to $f_{\rm kin} = f_{\rm col} = 1$ in the limit of large γ.

Figure 5.9 shows the dependence of the kinematic viscosity on the dimensionless mean free path λ.

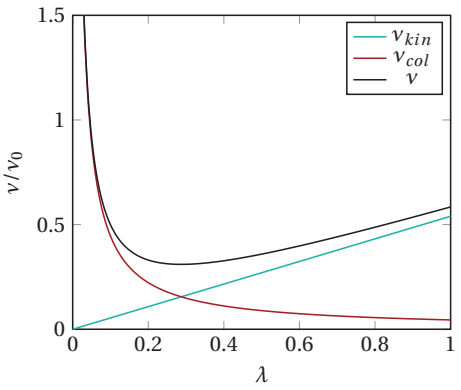

Figure 5.9: Kinematic viscosity of an SRD fluid for $\alpha = \pi/2$ and $\gamma = 5$ as a function of dimensionless mean free path λ. The kinetic contribution ν_{kin} is indicated in green, the collisional contribution ν_{col} in red, and the total kinematic viscocity ν in black.

Dynamic viscosity

The dynamic viscosity μ is related to the kinematic viscosity by $\mu = \rho_f \nu$, where $\rho_f = m\gamma/a_0^3$ is the fluid mass density. From Eqs. 5.51–5.54 it follows that the two contributions to the dynamic viscosity can be written in dimensionless form as:

$$\frac{\mu_{\text{kin}}}{\mu_0} = \frac{\lambda\gamma}{3} f^{\mu}_{\text{kin}}(\gamma,\alpha) \tag{5.55}$$

$$\frac{\mu_{\text{col}}}{\mu_0} = \frac{\gamma}{18\lambda} f^{\mu}_{\text{col}}(\gamma,\alpha). \tag{5.56}$$

where $\mu_0 = m/a_0 t_0 = \sqrt{mk_BT}/a_0^2$ is the unit of dynamic viscosity.

In contrast to the expressions for the diffusion of a fluid particle or a tagged particle, Eqs. (5.51) - (5.56) compare quantitatively to simulations over a wide range of parameters [34, 36, 55]. This is because the derivation of the SRD particle diffusion assumes molecular chaos and neglects density fluctuations. These are included in the more rigorous derivation of the viscosity. For the parameters we routinely used, i.e. $\lambda = 0.1, \alpha = \pi/2, \gamma = 5$, the collisional contribution to the viscosity dominates: $\nu_{kin} = 0.054\nu_0$ and $\nu_{\text{col}} = 0.45\nu_0$. This is typical for $\lambda \ll 1$, where ν and μ can be taken to a good first approximation by the collisional contribution only.

Difference between an SRD fluid and an ideal gas

It is instructive to compare the expressions derived in this section to what one would expect for simple gases, where, as famously first derived and demonstrated experimentally by Maxwell in the 1860's [44], the dynamic viscosity is independent of density. This result differs from the kinetic (gas-like) contribution to the viscosity in Eq. (5.55), because in a real dilute gas the mean-free path scales as $\lambda \propto 1/\gamma$, canceling the dominant density dependence in $\mu_{kin} \propto \gamma\lambda$. The same argument explains why the self-diffusion and kinematic viscosity of the SRD fluid are, to first order, independent of γ, while in a gas they would scale as $1/\gamma$. In SRD, the mean-free path λ and the density γ can be varied independently. Moreover, the collisional contribution to the viscosity adds

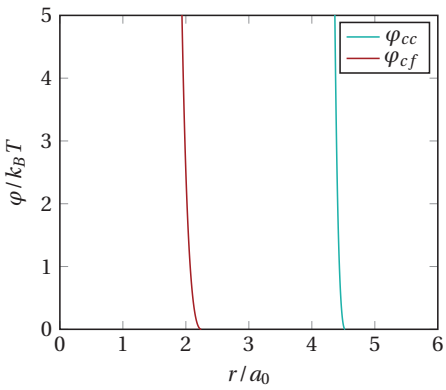

Figure 5.10: Colloid-colloid interaction φ_{cc} (Eq. 5.57, green) and colloid-fluid interaction φ_{cf} (Eq. 5.58, red) as a function of distance r. Here we show the case $\sigma_{cf} = 2a_0$, $\sigma_{cc} = 4.4a_0$, and $\epsilon_{cf} = \epsilon_{cc} = 2.5k_BT$, which we regularly employ.

a new dimension, allowing a much wider range of physical fluids to be modeled than just simple gases.

For example, the Schmidt number, which according to section 3.3 is the ratio of momentum diffusivity to mass-diffusivity, is close to unity in an ideal gas, whereas it is close to 1000 in liquid water. The Schmidt number of an SRD fluid can be made arbitrarily large by choosing an arbitrarily small λ.

Embedding colloidal particles in an SRD fluid

Colloidal particles can be embedded in an SRD fluid by a hybrid molecular dynamics (MD) scheme that couples a set of colloids to a bath of SRD particles [43, 51]. In these lectures we restrict ourselves to hard sphere-like colloids with steep interparticle repulsions, although attractions between colloids can easily be added on [48]. The colloid-colloid and colloid-fluid interactions, $\varphi_{cc}(r)$ and $\varphi_{cf}(r)$ respectively, are integrated via a normal MD procedure, while the fluid-fluid interactions are coarse-grained with SRD. Because the number of fluid particles vastly outnumbers the number of HS colloids, treating their interactions approximately via SRD greatly speeds up the simulation.

Colloid-colloid and colloid-solvent interactions

We can approximate pure hard-sphere colloidal interactions by a steep repulsive interactions of the Weeks-Chandler-Andersen (WCA) form with high exponents 48 and 24 [27]:

$$\varphi_{cc}(r) = \begin{cases} 4\epsilon_{cc}\left[\left(\frac{\sigma_{cc}}{r}\right)^{48} - \left(\frac{\sigma_{cc}}{r}\right)^{24} + \frac{1}{4}\right] & (r \le 2^{1/24}\sigma_{cc}) \\ 0 & (r > 2^{1/24}\sigma_{cc}). \end{cases} \tag{5.57}$$

Similarly, the colloid-fluid interaction takes the WCA form:

$$\varphi_{cf}(r) = \begin{cases} 4\epsilon_{cf}\left[\left(\frac{\sigma_{cf}}{r}\right)^{12} - \left(\frac{\sigma_{cf}}{r}\right)^{6} + \frac{1}{4}\right] & (r \le 2^{1/6}\sigma_{cf}) \\ 0 & (r > 2^{1/6}\sigma_{cf}). \end{cases} \tag{5.58}$$

Figure 5.10 shows these interactions for a specific set of parameters.

The mass m of a fluid particle is typically much smaller than the mass M_c of a colloid, so that the average thermal velocity of the fluid particles is larger than that of the colloid particles by a factor $\sqrt{M_c/m}$. For this reason the time-step Δt_{MD} is usually restricted by the fluid-colloid interaction (5.58), allowing fairly large exponents n for the colloid-colloid interaction $\varphi_{cc}(r) = 4\epsilon_{cc}\left[(\sigma_{cc}/r)^{2n} - (\sigma_{cc}/r)^{n} + 1/4\right]$. Above we have chosen $n = 24$, which makes the colloid-colloid potential fairly steep, more like hard-spheres, while still soft enough to allow the time-step to be set by the colloid-solvent interaction.

The positions and velocities of the colloidal spheres are propagated through the velocity Verlet algorithm (see section 2.7) [4] with a time step Δt_{MD}:

$$\mathbf{R}_i(t+\Delta t_{\mathrm{MD}}) = \mathbf{R}_i(t) + \mathbf{V}_i(t)\,\Delta t_{\mathrm{MD}} + \frac{\mathbf{F}_i(t)}{2M_c}\Delta t_{\mathrm{MD}}^2, \tag{5.59}$$

$$\mathbf{V}_i(t+\Delta t_{\mathrm{MD}}) = \mathbf{V}_i(t) + \frac{\mathbf{F}_i(t) + \mathbf{F}_i(t+\Delta t_{\mathrm{MD}})}{2M_c}\Delta t_{\mathrm{MD}}. \tag{5.60}$$

$\mathbf{R}_i$ and $\mathbf{V}_i$ are the position and velocity of colloid i, respectively. $\mathbf{F}_i$ is the total force on that colloid, exerted by the fluid particles, an external field, such as gravity, external potentials such as repulsive walls, as well as other colloids within the range of the interaction potential (5.57).

The positions $\mathbf{r}_i$ and velocities $\mathbf{v}_i$ of SRD particles are updated by similarly solving Newton's eqns. (5.44,5.45) every time-step Δt_{MD}, and with the SRD procedure of Eq. (5.47) every time-step δt_c.

Choosing both Δt_{MD} and δt_c as large as possible enhances the efficiency of a simulation. To first order, each timestep is determined by different physics, Δt_{MD} by the steepness of the potentials, and δt_c by the desired fluid properties, and so there is some freedom in choosing their relative values. We routinely use $\Delta t_c/\delta t_{\mathrm{MD}} = 4$ for our simulations of sedimentation [50], but other authors have used ratios of 50 [43] or even larger. In the next subsection we will revisit this question, linking the time-steps to various dimensionless hydrodynamic numbers and Brownian time-scales.

Stick and slip boundary conditions

Because the surface of a colloid is never perfectly smooth, collisions with fluid particles transfer angular as well as linear momentum. As demonstrated in Fig. 5.11, the exact molecular details of the colloid-fluid interactions may be very complex, and mediated via co- and counter-ions, grafted polymer brushes etc.... However, on the time and length-scales over which our hybrid MD-SRD method coarse-grains the solvent, these interactions can be approximated by *stick boundary conditions*: the tangential velocity of the fluid, relative to the surface of the colloid, is zero at the surface of the colloid [11, 28]. For most situations, this boundary condition should be sufficient, although in some cases, such as a non-wetting surface, large slip-lengths may occur [6].

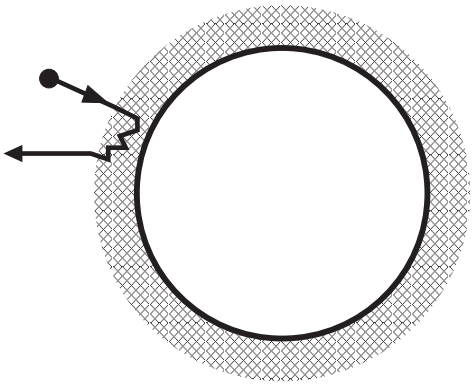

Figure 5.11: Schematic picture depicting how a fluid molecule interacts with a colloid, imparting both linear and angular momentum. Near the colloidal surface, here represented by the shaded region, there may be a steric stabilization layer, or a double-layer made up of co- and counter-ions. In SRD, the detailed manner in which a fluid particle interacts with this boundary layer is represented by a coarse-grained stick or slip boundary condition.

In SRD simulations, stick boundaries are best implemented in combination with sharp boundaries between the fluid and the wall or a colloidal particle. This way, during the streaming step, we can clearly identify SRD particles which find themselves in overlap with walls or colloidal particles. Stick boundary conditions may then be implemented by applying bounce-back rules to those particles, where both parallel and perpendicular components of the relative velocity of a fluid particle is reversed upon a collision with a surface [38]. Bounce-back rules were already discussed in section 2.5.

Alternatively, during the streaming step, stick boundaries can be modeled by a stochastic rule. As also discussed in section 2.5, after the collision, the relative tangential velocity v_t and relative normal velocity v_n of an SRD fluid particle may be taken from the distributions (in our units where $m = 1$ and $k_B T = 1$):

$$P(v_n) \propto v_n \exp\left(-v_n^2/2\right) \tag{5.61}$$

$$P(v_t) \propto \exp\left(-v_t^2/2\right), \tag{5.62}$$

so that the wall or colloidal particle also acts as a thermostat [29, 51]. We may argue that the stochastic rule of Eq. (5.61) is more like a real physical colloid – where fluid-surface interactions are mediated by steric stabilizing layers or local co- and counter-ion concentrations – than bounce-back rules are.

Up to this point we have discussed modifications during the streaming step (the molecular dynamics part) to achieve stick boundaries. Modifications of the collision (SRD rotation) step are also necessary. During the collision step collision cells may partially overlap with walls (because of the random grid shift) or colloidal particles (because of the curvature of the colloids). This results in a lower-than-average number of particles in such cells, leading to an altered fluid viscosity near the boundaries. To avoid spurious effects, it is sufficient to add *virtual particles* inside walls and objects such that they are filled with the same average number density (γ) as in the bulk fluid. For colloidal particles, this may be achieved by adding $\gamma\frac{4}{3}\pi\sigma_{cf}^3$ virtual particles at random locations in each colloidal particle. Each virtual particle is assigned a ve-

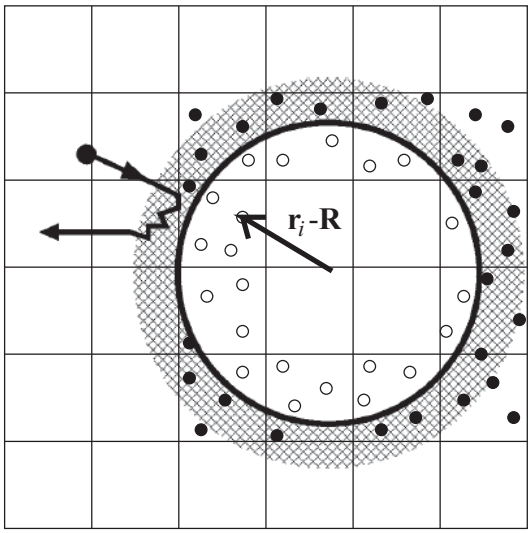

Figure 5.12: Same as Fig. 5.11, but now showing the SRD collision cells (square grid), real SRD fluid particles (filled dots) and virtual particles (open dots) which are added to the colloidal particle right before the SRD rotation collision step is executed. The virtual particles, with an average velocity given by the local colloid velocity plus a Maxwell-Boltzmann distributed random component, improve the stick-boundary conditions between the fluid and the colloidal particle.

locity equal to the local velocity of the colloidal particle plus a Maxwell-Boltzmann distributed random velocity (Figure 5.12).[10]

As is apparent from the above discussion, including colloidal particles with stick boundary conditions is not trivial. In these lectures we will therefore focus on the much simpler radial interactions such as those described in Eq. (5.58). These do not transfer angular momentum to a spherical colloid, and so induce effective *slip boundary conditions*.[11] For many of the hydrodynamic effects the difference with stick boundary conditions is quantitative, not qualitative, and also well understood.

Spurious depletion forces induced by the fluid

We would like to issue a warning that the additional fluid degrees of freedom may inadvertently introduce depletion forces between the colloids (Figure 5.13). Because the number density of SRD particles is much higher than that of the colloids, even a small overlap between two colloids can lead to enormous (effective) attractions.

For low colloid densities the equilibrium depletion interaction between any two colloids caused by the presence of the ideal fluid particles is given by [5]:

$$\Phi_{\rm depl}(d) = n_f k_B T \left[V_{\rm excl}(d) - V_{\rm excl}(\infty) \right], \tag{5.63}$$

where $n_f = \gamma / a_0^3$ is the number density of fluid particles and $V_{\rm excl}(d)$ is the (free) volume excluded to the fluid by the presence of two colloids separated by a distance d.

[10] Filling static walls with virtual particles is often done more simply by adding $(\gamma - n)$ Maxwell-Boltzmann distributed velocities during the calculation of the centre-of-mass velocity in a wall cell, where n is the actual number of particles in that particular wall cell.

[11] For slip boundaries it is not necessary to add virtual particles because there is no shear gradient near the boundaries. The viscosity very close to the boundaries is therefore less important than in the case of stick boundaries.

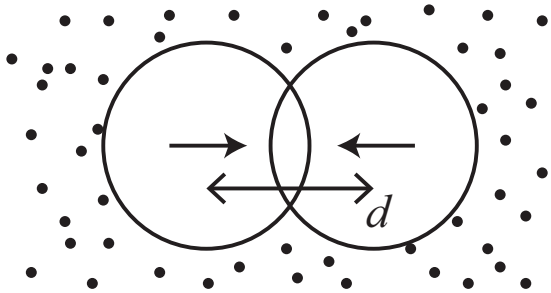

Figure 5.13: Colloidal particles (spheres) exclude the fluid particles (points) from their volume. If two colloidal particles partly overlap, the (osmotic) pressure forces from the fluid particles are unbalanced, leading to an effective force pushing the two colloidal particles together. Note that depletion forces are equilibrium (thermodynamic) forces.

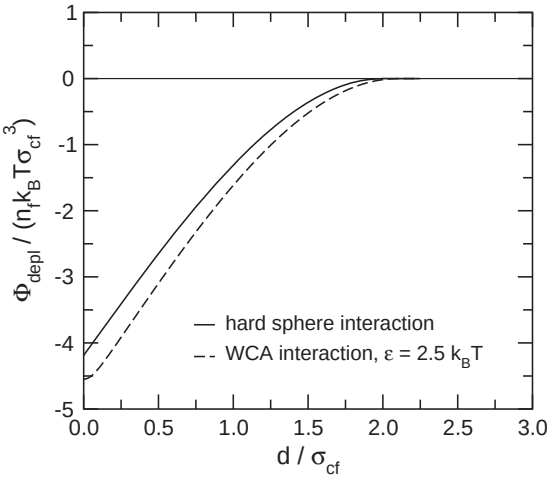

Figure 5.14: Effective depletion potentials induced between two colloids by the SRD fluid particles, for hard-sphere fluid-colloid (solid line) and WCA fluid-colloid (dashed line) interactions taken from Eq. (5.58). The interparticle distance d is measured in units of σ_{cf}. Whether these attractive potentials have important effects or not depends on the choice of diameter σ_{cc} in the bare colloid-colloid potential (5.57) [51].

The latter is given by

$$V_{\rm excl}(d) = \int {\rm d}^3\mathbf{r}\left\{1-\exp\left[-\beta\varphi_{cf}\left(\mathbf{r}-\mathbf{r}_1\right)-\beta\varphi_{cf}\left(\mathbf{r}-\mathbf{r}_2\right)\right]\right\}, \tag{5.64}$$

where $|\mathbf{r}_1-\mathbf{r}_2| = d$ and $\beta = 1/(k_BT)$. An example is given in Fig. 5.14 where we have plotted the resulting depletion potential for the colloid-solvent interaction (5.58), with $\epsilon_{cc} = \epsilon_{cf} = 2.5k_BT$ as routinely used in our simulations, as well as the depletion interaction resulting from a truly hard-sphere colloid-solvent interaction. The latter can easily be calculated analytically, with the result

$$\Phi^{\rm HS}_{\rm depl}(d) = -n_f k_B T \frac{4}{3}\pi\sigma_{cf}^3\left(1-\frac{3d}{4\sigma_{cf}}+\frac{d^3}{16\sigma_{cf}^3}\right) \qquad (\text{for } d<2\sigma_{cf}). \tag{5.65}$$

For the pure hard-sphere interactions, one could take $\sigma_{cc} \geq 2\sigma_{cf}$ and the depletion forces would have no effect. But for interactions such as those used in Eqs. (5.57) and (5.58), the softer repulsions mean that inter-colloid distances less than σ_{cc} are regularly sampled. A more stringent criterion of σ_{cc} must be therefore be used to avoid spurious depletion effects.

Since the depletion potentials can be calculated analytically, one might try counteracting them by introducing a compensating repulsive potential of the form: $\Phi_{\rm comp}$ =

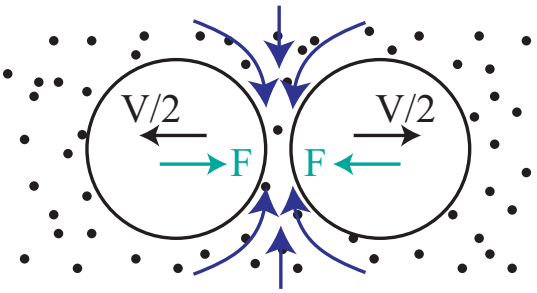

Figure 5.15: When colloidal particles move apart with relative velocity V, fluid must flow into the freed up space. At very small gap widths between the colloidal surfaces this leads to strong lubrication forces F which oppose the direction of V. Note that lubrication forces are non-equilibrium forces that scale with the relative velocity V.

$-\Phi_{\text{depl}}$ between the colloids. However, there are three problems with this approach: Firstly, at higher colloid packing fractions, three and higher order interactions may also need to be added, and these are very difficult to calculate. Secondly, the depletion interactions are not instantaneous, and in fact only converge to their equilibrium average algebraically in time [60]. While this is not a problem for equilibrium properties, it will introduce errors for non-equilibrium properties. Finally, when external fields drive the colloid, small but persistent anisotropies in the solvent density around a colloid may occur [21]. Although these density variations are (and should be) small, the resulting variations in depletion interactions can be large.

To avoid these problems, we routinely choose the colloid-fluid interaction range σ_{cf} slightly below half the colloid diameter $\sigma_{cc}/2$. More precisely, we ensure that the colloid-colloid interaction equals $2.5k_BT$ at a distance d where the depletion interactions have become zero, i.e., at a distance of twice the colloid-solvent interaction cut-off radius (leading to $\sigma_{cc} \approx 2.2\sigma_{cf}$). Smaller distances will consequently be rare, and adequately dealt with by the compensation potential. This solution may be a more realistic representation anyhow, since in practice for charge and even for sterically stabilized colloids, the effective colloid-colloid diameter σ_{cc} is expected to be larger than twice the effective colloid-fluid diameter σ_{cf}. This is particularly so for charged colloids at large Debye screening lengths.

Lubrication forces

When two surfaces approach one another, they must displace the fluid between them, while if they move apart, fluid must flow into the space freed up between the surfaces (Figure 5.15). At very short inter-surface distances, this results in so-called lubrication forces, which are repulsive for colloids approaching each other, and attractive for colloids moving apart [19, 28, 56]. These forces are expected to be particularly important for driven *dense* colloidal suspensions [59].

An additional advantage of our choice of diameters σ_{ci} above is that more fluid particles will fit in the space between two colloids, and consequently the lubrication forces will be more accurately represented. It should be kept in mind that for colloids, the exact nature of the short-range lubrication forces will depend on physical details of the surface, such as its roughness, or presence of a grafted polymeric stabilizing layer [46]. For perfectly smooth colloids, analytic limiting expressions for the lubrication forces

can be derived [28], showing a divergence at short distances. We have confirmed that SRD resolves these lubrication forces down to surprisingly low interparticle distances, of the order of half a collision cell size a_0 [52]). But, at some point this will break down, depending of course on the choice of simulation parameters (such as σ_{cf}/a_0, λ, and γ), as well as the details of the particular type of colloidal particles that one wishes to model [46]. An explicit analytic correction could be applied to properly resolve these forces for very small distances, as has been implemented for Lattice Boltzmann [49]. However, in most application with radial interactions between the colloids and SRD fluid, σ_{cf} is chosen small enough for SRD to sufficiently resolve lubrication forces. The lack of a complete divergence at very short distances may be a better model of what happens for real colloids anyway. For dense suspensions under strong shear, explicit lubrication force corrections as well as other short-ranged colloid-colloid interactions arising from surface details such as polymer coats will almost certainly need to be put in by hand, see e.g. ref. [46] for further discussion of this subtle problem.

When the dust has settled: choosing the parameters

Suppose that finally, after some hard work, you have programmed a code that simulates an SRD fluid with embedded colloidal particles. The question you will probably ask is: how to choose the parameters of the model? An overview of all free parameters is given in Table 5.1. We will now deal with each free parameter and discuss the reasons for common choices. We end with a discussion on how to relate the simulation results to those of real colloidal suspensions.

Average number of particles per cell γ

The average number of particles per cell γ should not be too low, because two or more particles per cell are needed to actually have collisions between particles. Here we should take into account that an SRD fluid acts in many ways as an ideal gas. In particular, it has the same number density fluctuations as an ideal gas ($\Delta\gamma = \sqrt{\gamma}$). A too low γ will make the collisions very inefficient and cause a (too) slow relaxation of the particle velocities to thermal equilibrium.

The higher the number of particles in a cell, the lower the relative number fluctuations (and also the smoother the flow fields can be visualised, if this is the goal of the simulation). However, a very large number of particles per cell means a very high computational effort.

A good compromise between efficient collisions and low computational effort is reached for values in the range $3 < \gamma < 10$.

Collision time interval δt_c

The collision time interval δt_c should not be too high, because this will lead to a mean free path larger than the cell size a_0; it does not make sense to let the particles stream more than a collision cell size in between collisions. Since with our choice of unit

parameters, the mean free path is essentially the same as the collision time interval, Eq. (5.48), we routinely choose $\delta t_c \leq 1$ (in our time unit t_0). Conversely, Eq. (5.52) shows that high viscosities may be achieved by choosing low values of δt_c, but this may come at the computational cost of having to perform many SRD rotation collisions per t_0.

For liquid-like behaviour (as opposed to gas-like behaviour) it is important to achieve a high Schmidt number. Although for water the Schmidt number is approximately 1000, for most applications $\mathrm{Sc} \gtrsim 10$ is already sufficiently liquid-like. This is achieved when $\delta t_c < 0.1$.

Depending on the application, values in the range $0.01 < \delta t_c \leq 1$ are routinely chosen.

Rotation angle α

The rotation angle α influences the viscosity too, see Eqs. (5.51) and (5.52), but in practice not as strongly as the collision time interval. At first sight extreme values of α near zero would correspond to a very high kinetic viscosity, but this corresponds to a situation where the collisions between particles are again very inefficient, leading to too slow relaxation of the particle velocities to thermal equilibrium (just as in the case of a very low number of particles per cell). Using the other extreme of α near π also creates problems because in that case relative velocities are nearly exactly inverted. Admissable values are in the range $\pi/10 < \alpha < 9\pi/10$ (or in degrees: $20^\circ < \alpha < 160^\circ$).

The rotation operation is much simpler for $\alpha = \pi/2$ than for other angles, with an associated increase in computational efficiency. Therefore, in practice, $\alpha = \pi/2$ is a popular choice.

Colloid-fluid and colloid-colloid interaction parameters σ_{cf}, σ_{cc}, ϵ_{cf} and ϵ_{cc}

The values of ϵ_{cf} and ϵ_{cc} should be such that we may indeed identify σ_{cf} and σ_{cc} as *the* colloid-solvent radius and colloid diameter. This is rather arbitrary for non-hard-sphere interactions, but as a rule of thumb we may define the above radius and diameter as the distance at which the probability of encountering an SRD fliud particle or another colloidal particle, respectively, has decayed to 10% of the probability in the bulk (far away). Using Eqs. (5.57) and (5.58) in the Boltzmann distribution, this leads to $\epsilon_{cf} = \epsilon_{cc} = -\ln(0.1) \approx 2.3 k_B T$. To be on the safe side, we routinely use $2.5 k_B T$.

Next we consider the colloid-solvent radius σ_{cf}. In SRD the hydrodynamic fields are accurately resolved to a scale of the order of the collision cell size (typically down to $0.5a_0$ for small mean free paths). An important question is therefore: how large should we choose the colloidal particle relative to the collision cell size. The answer, as always, depends on the amount of accuracy we desire. Simulations have shown [51] that the flow field around, and drag force on, a moving colloidal particle is already resolved with typical errors of 10% or less for $\sigma_{cf} = 2a_0$. The error decreases to the order of 2% for $\sigma_{cf} = 4a_0$. For the same number of colloidal particles and the same solid volume fraction, the number of SRD particles needed in the simulation scales as σ_{cf}^3 (in 3D), so in practice values in the range $2a_0 < \sigma_{cf} < 4a_0$ are used.

Finally, to avoid depletion forces, as discussed previously, the colloid-colloid diameter σ_{cc} should be chosen larger than $2\sigma_{cf}$. How much larger depends on the range of the nearly-hard-sphere colloid-fluid interactions. Figure 5.14 shows that for our choice of radial interactions, $\sigma_{cc} = 2.2\sigma_{cf}$ is a good choice.

Relating the simulation results to real colloidal suspensions: time scales

For realistic dynamical behaviour of the colloidal suspension, it is important that certain characteristic time scales are in the correct order [19]. For example, for a 1 micron colloidal particle in water at room temperature, we can identify the following characteristic time scales:

- The fluid velocity time scale at which molecular velocities of the fluid decorrelate: $\tau_f = 10^{-14}$ s.
- The Fokker-Planck time scale at which forces on a colloidal particle decorrelate: $\tau_{FP} = 10^{-13}$ s.
- The sonic time scale for sound to propagate over a colloid radius: $\tau_{cs} = 10^{-10}$ s.
- The kinematic time scale for momentum to diffuse over a colloid radius: $\tau_\nu = 10^{-6}$ s.
- The diffusive time scale for the colloid to self-diffuse over a colloid radius: $\tau_D = 10^0$ s.

For correct colloidal hydrodynamics, these time scales should be separated as:

$$\tau_f \approx \tau_{FP} < \tau_{cs} < \tau_\nu \ll \tau_D. \tag{5.66}$$

The above numbers show that for real colloidal particles, the range of relevant time scales is enormous, spanning at least 14 orders of magnitude. Clearly it would be impossible to bridge all the time-scales of a physical colloidal system – from the molecular τ_f to the mesoscopic τ_D – in a single simulation. Thankfully, it is not necessary to exactly reproduce each of the different time-scales in order to achieve a correct coarse-graining of colloidal dynamics. As long as they are clearly separated, and in the right order, the correct physics should still emerge.

Regarding the fluid velocity time scale, in SRD the effect of the collisions calculated in an average way every time-step δt_c. The time-scale τ_f on which the velocity correlations decay can be quite easily calculated from a random-collision approximation. Following [55]: $\tau_f \approx -\lambda t_0 / \ln[1 - \frac{2}{3}(1-\cos\alpha)(1-1/\gamma)]$. For our usual parameters, $\lambda = 0.1, \alpha = \frac{1}{2}\pi, \gamma = 5$, we find $\tau_f \approx 0.76\delta t_c = 0.076 t_0$.

Regarding the Fokker-Planck time scale, we expect the Fokker Planck time τ_{FP} to scale as δt_c, since this is roughly equivalent to the time τ_f over which the fluid velocities will have randomized. Simulations show that this is indeed the case [51].

Regarding the sonic time scale, for our choice of units $c_s = \sqrt{5/3}a_0/t_0$, so that the sonic time scale reduces to

$$\tau_{cs} \approx 0.775\frac{\sigma_{cf}}{a_0}, \tag{5.67}$$

which is independent of λ or γ.

In the limit of small λ, the ratio of the kinematic time τ_ν to δt_c can be simplified to:

$$\frac{\tau_\nu}{\delta t_c} \approx 18\frac{\sigma_{cf}^2}{a_0^2} \tag{5.68}$$

so that the condition $\tau_f, \tau_{FP} \ll \tau_\nu$ is very easy to fulfill. Furthermore, under the same approximations, the ratio $\tau_\nu/\tau_{cs} \approx 23\sigma_{cf}\lambda$ so that for λ too small the kinematic time becomes faster than the sonic time. For our usual parameters $\lambda = 0.1, \alpha = \frac{1}{2}\pi, \gamma = 5, \sigma_{cf} = 2a_0$, we find $\tau_\nu/\tau_{cs} \approx 5$.

The colloid diffusion coefficient is directly related to the friction by the Einstein relation (5.14). If we assume that $\lambda \ll 1$ and, for simplicity, that the friction ζ is given by the Stokes law,[12] then the diffusion time scales as:

$$\tau_D = \frac{\sigma_{cf}^2}{D_{\text{col}}} \approx \frac{6\pi\mu\sigma_{cf}^3}{k_B T} \approx \frac{\gamma}{\lambda}\left(\frac{\sigma_{cf}}{a_0}\right)^3 t_0. \tag{5.69}$$

It is instructive to examine the ratio of the diffusion time τ_D to the kinematic time τ_ν:

$$\frac{\tau_D}{\tau_\nu} = \frac{\nu}{D_{\text{col}}} \approx 0.06\frac{\gamma}{\lambda^2}\left(\frac{\sigma_{cf}}{a_0}\right). \tag{5.70}$$

In general we advocate keeping λ small to increase the Sc number $\text{Sc} = \nu/D_f$, and since another obvious constraint is $D_{\text{col}} \ll D_f$, there is not too much difficulty achieving the desired separation of time-scales $\tau_\nu \ll \tau_D$.

As a concrete example of how these time-scales are separated in a simulation, consider the parameters used routinely in our work: for $\alpha = \frac{1}{2}\pi, \gamma = 5$, $\lambda = 0.1$ and $\sigma_{cs} = 2a_0$, we find $\tau_f = 0.076t_0$, $\tau_{FP} = 0.09t_0$, $\tau_{cs} = 1.4t_0$, $\tau_\nu = 8t_0$ and $\tau_D = 200t_0$. More generally, what the analysis of this section shows is that obtaining the correct hierarchy of time-scales,

$$\tau_f \approx \tau_{FP} < \tau_{cs} < \tau_\nu \ll \tau_D$$

[12] Because in particle-based mesoscale methods a relatively low number of fluid particles is in contact with each colloidal particle, there appears also another source of friction, which may be termed Enskog friction ζ_E. This is the friction experienced by a colloidal particle moving through a non-hydrodynamic thermal bath of the same number density as the SRD fluid. The Stokes and Enskog frictions can be added (approximately) in parallel, i.e. $1/\zeta = 1/\zeta_S + 1/\zeta_E$ [51]. For real colloidal particles, the number of fluid molecules in contact with a colloidal particles is much higher, and consequently $\zeta_E \gg \zeta_S$. The Enskog friction can then be neglected: $\zeta \approx \zeta_S$.

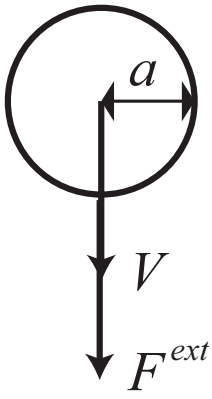

Figure 5.16: Simplified sketch of a colloidal particle of radius a experiencing an external force F^{ext} such as gravity. The colloidal particle will not only perform random Brownian motion, but as a consequence of the external field also develop an average velocity V in the direction of the external field.

is virtually guaranteed once the parameters are chosen as suggested in the previous sections.

To achieve a mapping between the simulations and a real colloidal system, we need to determine the real scales of the three units, a_0, m and $k_B T$, of our simulations. Usually a_0 is set by equating the simulated colloidal diameter to the real colloidal diameter, and $k_B T$ can be set to its real value. Finally, the SRD particle mass m is determined by setting the simulated self-diffusion coefficient of a single colloidal particle equal to that in the real system, making use of the unit of diffusion $D_0 = a_0\sqrt{k_B T/m}$ defined in Table 5.1.[13]

5.8 Colloidal suspensions in external fields and flows: importance of dimensionless numbers

In many (engineering) applications, colloidal particles are either subjected to external force fields – such as gravity, centrifugation, electric or magnetic fields – or being forced to flow as part of a flowing colloidal suspension. In all these cases, the colloidal particles are not only transported through random diffusive motion, but also by convective motion. Figure 5.16 shows a simplified example of a colloidal particle convecting in the direction of the external field (e.g. sedimentation in the direction of gravity).

No matter by which mesoscale method the fluid is modelled, it is important to be aware of dimensionless numbers characterising this diffusive and convective processes inside the colloidal suspension. We already introduced these dimensionless numbers in chapter 3. Here we will focus on the peculiarities of mesoscale methods with respect to the most important dimensionless numbers.

[13] When using the Stokes-Einstein equation, this is equivalent to equating the dynamic viscosity of the fluid. Note that it is *not* the fluid mass density ρ that is compared; because colloidal systems are in the low-Re regime, the mass density of the fluid is (mostly) irrelevant. Connected to this, gravity forces acting on the fluid should be avoided as much as possible because of potential problems with the high compressibility of an SRD fluid. For sedimentation problems such as treated in the next section, it is better to apply the net buoyant forces on the colloidal particles.

The Mach number in colloidal suspensions

The Mach number measures the ratio between V, the speed of solvent or colloid flow, and c_s, the speed of sound in the fluid,

$$\mathrm{Ma} = \frac{V}{c_s}. \tag{5.71}$$

The Ma number measures compressibility effects [26] since sound speed is related to the compressibility of a liquid. Because c_s in many liquids is of order 10^3 m/s, the Ma numbers for physical colloidal systems are extremely small under normally achievable flow conditions. Just as for the Schmidt number, however, particle based coarse-graining schemes drastically lower the Ma number. The fluid particle mass m is typically much greater than the mass of a molecule of the underlying fluid, and moreover, due to the lower density, collisions also occur less frequently. These effects mean that the speed of sound is much lower in a coarse-grained system than it is in the underlying physical fluid. Or, in other words, particle based coarse-graining systems are typically much more compressible than the solvents they model.

Ma number effects typically scale with Ma^2 [26], and so the Ma number does not need to be nearly as small as for a realistic fluid to still be in the correct regime of hydrodynamic parameter space. This is good, because to lower the Ma number, one would need to integrate over longer fluid particle trajectories to allow, for example, a colloidal particle to flow over a given distance, making the simulation more computationally expensive. So there is a compromise between small Ma numbers and computational efficiency.

It is best practice to limit the Ma numbers to values such that $\mathrm{Ma} \leq 0.1$, but it might be possible, in some situations, to double or triple that limit without causing undue error. For example, incompressible hydrodynamics is used for aerodynamic flows up to such Ma numbers since the errors are expected to scale as $1/(1-\mathrm{Ma}^2)$ [26].

Specifically for SRD, when working in units of m and $k_B T$, the only way to keep the Ma number below this upper limit is to restrict the maximum flow velocity to $V \leq 0.1 c_s \approx 0.13 a_0/t_0$. The flow velocity itself is, of course, determined by the external force fields or imposed flow velocities.

The Reynolds number in colloidal suspensions

The Reynolds number determines the relative importance of inertial over viscous forces. For a colloidal particle of radius a moving with average velocity V, it can be expressed as:

$$\mathrm{Re} = \frac{Va}{\nu}. \tag{5.72}$$

For a spherical particle in a flow, the following heuristic argument helps clarify the physics behind the Reynolds number: If the *Stokes time*

$$t_S = \frac{a}{V}, \tag{5.73}$$

it takes a particle to move over its own radius is about the same as the *kinematic time*

$$\tau_\nu = \frac{a^2}{\nu} = \mathrm{Re}\ t_S \tag{5.74}$$

it takes momentum to diffuse over that distance, i.e. Re=$\tau_\nu / t_S \approx 1$, then the particle will feel vorticity effects from its own motion a distance a away, leading to non-linear inertial effects. Since hydrodynamic interactions can decay as slowly as a/r, their effects can be non-negligible. If, on the other hand, Re $\ll 1$, then the particle will only feel very weak hydrodynamic effects from its own motion.

Exactly when inertial finite Re effects become significant depends on the physical system under investigation. For a single spherical particle, inertial effects which induce a non-linear dependence of the friction ζ on the velocity V start to become noticeable for a particle Reynolds number of Re ≈ 1 [26], while deviations in the symmetry of the streamlines around a rotating sphere have been observed in calculations for Re ≈ 0.1 [47].

For typical colloidal suspensions, where the particle diameter is on the order of a few μm down to a few nm, the particle Reynolds number is rarely more than 10^{-3}. In this Stokes regime viscous forces dominate and inertial effects can be completely ignored. The Navier-Stokes equations can be replaced by the linear Stokes equations [26] so that analytic solutions are easier to obtain [28]. However, some of the resulting behavior is non-intuitive for those used to hydrodynamic effects on a macroscopic scale. For example, as famously explained by Purcell in a talk entitled "*Life at low Reynolds numbers*" [54], many simple processes in biology occur on small length scales, well into the Stokes regime. The conditions that bacteria, typically a few μm long, experience in water are more akin to those humans would experience in extremely thick molasses. Similarly, colloids, polymers, vesicles and other small suspended objects are all subject to the same physics, their motion dominated by viscous forces. For example, if for a colloid sedimenting at 1 μm/s, gravity were instantaneously turned off, then the Stokes equations suggest that it would come to a complete halt in a distance significantly less than one Å, reflecting the irrelevance of inertial forces in this low Re number regime. It should be kept in mind that when the Stokes regime is reached because of small length scales (as opposed to very large viscosities such as those found for volcanic lava flows), then thermal fluctuations are also important. These will drive diffusive behavior [9]. In many ways SRD is ideally suited for this regime, assuming that one can indeed reach low Re numbers, because the thermal fluctuations are naturally included [43].

Specifically for SRD, from Eqs. (5.51) – (5.52) and Eq. (5.72), it follows that the Reynolds number for a colloid of hydrodynamic radius $a \approx \sigma_{cf}$ can be written as:

$$\mathrm{Re} \quad = \quad \sqrt{\frac{5}{3}}\,\mathrm{Ma}\left(\frac{\sigma_{cf}}{a_0}\right)\left(\frac{\nu_0}{\nu}\right) \tag{5.75}$$

where the Ma number is defined in Eq. (5.71). Equating the hydrodynamic radius a to σ_{cf} from the fluid-colloid WCA interaction of Eq. (5.58) is not quite correct, but is a good enough approximation.

In order to keep the Reynolds number low, one must either use small particles, or very viscous fluids, or low velocities V. The latter condition is commensurate with a low Ma number, which is also desirable. For small enough mean free path λ (and ignoring $f^{\mu}_{\rm col}(\gamma,\alpha)$) the Re number then scales as

$$\mathrm{Re} \approx 23\ \mathrm{Ma}\ \frac{\sigma_{cf}}{a_0}\ \lambda. \tag{5.76}$$

Again, we see that a smaller mean-free path λ, which enhances the collisional viscosity, also helps bring the Re number down. This parameter choice is also consistent with a larger Sc number. Thus larger Sc and smaller Ma numbers both help keep the Re number low for SRD. In principle a large viscosity can also be obtained for large λ, which enhances the kinetic viscosity, but this choice also lowers the Sc number and raises the Knudsen number, which, as we will see in the next sub-section, is not desirable.

Just as was found for the Ma number, it is relatively speaking more computationally expensive to simulate for low Re numbers because the flow velocity must be kept low, which means longer simulation times are necessary to reach time-scales where the suspended particles or fluid flows have moved a significant distance. We therefore usually compromise and keep $\mathrm{Re} \lesssim 0.2$ [37,51]. For many situations related to the flow of colloids, this should be sufficiently stringent.

The Knudsen number in colloidal suspensions

The Knudsen number Kn is the ratio between the mean free path λ_{free} of the molecules and a characteristic length scale, which in this case is the colloidal radius a,

$$\mathrm{Kn} = \frac{\lambda_{free}}{a}. \tag{5.77}$$

For rarified gas flows the Knudsen number is large (> 10), and continuum Navier-Stokes equations completely break down.

However, the continuous phase in a colloidal suspension is a liquid. The mean free path of most liquids is quite small. For water at standard temperature and pressure $\lambda_{\rm free} \approx 3$ Å. Just as found for the other dimensionless numbers, coarse-graining typically leads to larger Kn numbers because of the increase of the mean-free path. Making the Kn number smaller also typically increases the computational cost because a larger number of collisions need to be calculated. In mesoscale simulations, it is important to keep $\mathrm{Kn} \leq 0.05$. This rough criterion is based on the observation that for small Kn numbers the friction coefficient on a sphere is expected to be decreased by a factor $1 - \alpha\ \mathrm{Kn}$, where α is a material dependent constant of order 1 [28], so that we expect Kn number effects to be of the same order as other coarse-graining errors. There are two ways to achieve small Kn numbers: one is by by increasing the modeled radius a of the colloidal particle relative to the intrinsic coarse-graining scale (e.g. σ_{cf}/a_0 in SRD), the other is by decreasing the mean free path λ_{free}. The second condition is commensurate with a large Sc number or a small Re number.

The Peclet number in colloidal suspensions

The Peclet number Pe measures the relative strength of convective transport to diffusive transport. For example, for a colloid of radius a, traveling at an average velocity V, the Pe number is defined as:

$$\mathrm{Pe} = \frac{Va}{D_{\mathrm{col}}}, \qquad \text{(general)} \tag{5.78}$$

where D_{col} is the colloid diffusion coefficient. For colloidal particles in an external field we can use the Einstein relation, $D_{col} = k_B T/\zeta$, and the relation between external force and average velocity in steady-state, $V = F^{ext}/\zeta$, to rewrite the Peclet number as:

$$\mathrm{Pe} = \frac{F^{ext} a}{k_B T}, \qquad \text{(external field)} \tag{5.79}$$

which shows that the Peclet number may also be interpreted as the ratio of energy $F^{ext} a$ gained by a colloidal particle by moving one radius a in the direction of the external field and the thermal energy $k_B T$.

Alternatively, just as for the Re number, the Pe number can be interpreted as a ratio of a diffusive to a convective time-scale, but now the former time-scale is not for the diffusion of momentum but rather it is given by the *colloid diffusion time*

$$\tau_D = \frac{a^2}{D_{\mathrm{col}}} \tag{5.80}$$

which measures how long it takes for a colloid to self-diffuse over a distance a. The Pe number can then be written as:

$$\mathrm{Pe} = \frac{\tau_D}{t_S}. \qquad \text{(general)} \tag{5.81}$$

If $\mathrm{Pe} \gg 1$ then the colloid moves convectively over a distance much larger than its radius a in the time τ_D that it diffuses over that same distance. Brownian fluctuations are expected to be less important in this regime. For $\mathrm{Pe} \ll 1$, on the other hand, the opposite is the case, and the main transport mechanism is diffusive (note that on long enough time-scales ($t > \tau_D/\mathrm{Pe}^2$) convection will always eventually "outrun" diffusion [9]). It is sometimes thought that for low Pe numbers hydrodynamic effects can be safely ignored, but this is not always true. For example, we found that the reduction of average sedimentation velocity with particle volume fraction, famously first explained by Batchelor [7], is independent of Pe number down to $\mathrm{Pe} = 0.1$ at least [50].

The highest Pe number achievable in particle-based mesoscale simulations is limited by the constraints on the Ma and Re numbers. For example the Ma number sets an upper limit on the maximum Pe number by limiting V.

Specifically for SRD, from Eqs. (5.72) and (5.78), it follows that the Peclet number can be re-written in terms of the Reynolds number as

$$\mathrm{Pe} = \frac{\nu}{D_{\mathrm{col}}}\mathrm{Re} \approx 6\pi\gamma\left(\frac{\nu}{\nu_0}\right)^2\left(\frac{\sigma_{cf}}{a_0}\right)\mathrm{Re} \tag{5.82}$$

where we have approximated $D_{\text{col}} \approx k_B T/(6\pi\eta\sigma_{cf})$. This shows that for a given constraint on the Re number, increasing γ or σ_{cf} increases the range of accessible Pe numbers. Similarly, when the kinematic viscosity is dominated by the collisional contribution, decreasing the dimensionless mean-free path λ will also increase the maximum Pe number allowed since $\text{Pe}^{max} \sim \lambda^{-2}\text{Re} \sim \lambda^{-1}\text{Ma}$.

5.9 Limitations of mesoscopic simulation methods

Many different particle-based methods exist to deal with the low-Reynolds number dynamics of mesoscale particles such as colloidal particles. Which one to choose depends on the system of interest, because each method has its own limitations. Let us summarize the range of applicability and limitations of each of the methods we have discussed in more detail.

- **Langevin dynamics** (LD) simulations neglect hydrodynamic interactions, and are therefore limited to situations of either very low solid volume fractions ($\phi < 10^{-4}$) or when the particles are so close that they significantly overlap in their direct interaction range. Being a second-order algorithm (including velocities), larger particle frictions necessitate lower time steps, making Langevin dynamics only suitable for situations with relatively low friction forces such as large molecules or very small colloids of less than a nanometre.

- **Brownian dynamics** (BD) simulations also neglect hydrodynamic interactions, and are therefore also limited to very low solid volume fractions or significantly overlapping particles. Being a first-order algorithm (it is assumed that velocities are equilibrated before the particle has moved a fraction of its size), larger particle frictions allow for larger time steps, making Brownian dynamics simulations very suitable for situations with very high friction.

- **Stokesian dynamics** (SD) simulations are the equivalent of Brownian dynamics, but including hydrodynamic interactions between the particles and is therefore much more generally applicable. However, because of its complexity and high computational costs, Stokesian dynamics is limited to a few hundred particles. Moreover, being based on analytical expressions for the hydrodynamic interactions in an unbounded medium, it is very difficult to study the dynamics of colloidal suspensions confined by walls or other boundaries.

- **Dissipative particle dynamics** (DPD) is a particle-based method for mesoscale fluids, which is relatively easy to implement in a code. Colloidal particles can be embedded as extra large DPD particles or by freezing assemblies of DPD particles. However, there are two limitations. First, the repulsive interactions between the particles cause an ordering in the fluid structure, especially near walls and colloidal particles, that may be felt as oscillating conservative interactions between the colloidal particles when their surfaces are closer than a couple of

r_c apart. Therefore DPD cannot be used in situations where lubrication forces are important (dense solutions of nearly hard spheres). Second, to be in the hydrodynamic limit, colloidal particles need to be much larger than the DPD fluid particles. Because the interactions between the DPD particles are still based on pair interactions, the high computational effort limits the number of colloidal particles again to a few hundred.

- **Multiparticle collision dynamics** (MPCD), and its specific implementation of **stochastic rotation dynamics** (SRD), is also a particle-based method, but the collisions between the fluid particles are handled much more efficiently (compared to DPD) in collision cells. Colloidal particles can be embedded through repulsive potential energy functions or by reflecting boundary conditions. Because the thermodynamic properties of an MPCD fluid are equal to that of an ideal gas, it is much more compressible than the real fluid it represents. One should therefore always be aware of the Mach number to avoid density fluctuations in the system. Also, because the particles have inertia, it is sometimes a challenge to keep the Reynolds number down when colloidal particles are driven by an external field or flow (these inertial limitations also apply to DPD). The Mach and Reynolds number limitations also pose a limit on the largest Peclet number that can be reached. In practice, MPCD is ideally suited for driven colloidal systems with Peclet numbers ranging as $0 < \mathrm{Pe} < 30$ in systems containing thousands of colloidal particles and tens of millions of fluid particles.

It is important to emphasize that thermal fluctuations are a key ingredient of *all* the above methods. This is precisely the definition of the mesoscale regime. When dealing with macroscopic particles, or if one is interested in average flowfields or average drag relations, the thermal fluctuations are sometimes more of a nuisance than an aid, because a lot of averaging is required before the effects of thermal fluctuations have been averaged out. In that case it may be better to use direct numerical simulations such as Lattice Boltzmann (in its original form, i.e. without thermal fluctuations) or finite volume of finite element discretisations of the Navier-Stokes equations, combined with immersed particles, as will be briefly discussed in the next chapter.

5.10 Practicum: Colloidal sedimentation

In this practicum you will study the behaviour of nearly hard sphere colloidal particles sedimenting in a slit using stochastic rotation dynamics for the fluid. You will measure the sedimentation velocity as a function of solid volume fraction.

CHAPTER 6

THE MACROSCOPIC WORLD

6.1 Chapter objectives

Through the course of this chapter, you will accomplish the following:

- You will learn about the main particle-based simulation models for granular systems
- You will learn about a common contact model used in simulations of granular systems
- You will learn to distinguish between resolved and unresolved coupling to fluid flow
- You will learn about limitations of macroscopic particle-based methods

6.2 Introduction to granular systems

When particles are 100 micrometer in size or larger, Brownian motion due to thermal fluctuations is no longer important: they are *athermal*. Such particles are often referred to as *granular* particles. These are the particles that we are most familiar with: flowing sand, rock avelanches, emptying hoppers filled with grains, pneumatic conveying of particles and powders, and mixing and segregation of particles when they are transported and shaken. The dynamics of these systems are dominated by gravity and friction effects. Friction means that, without further perturbations such as shaking or a fluidisation by a gas flow, the particles will quickly come to complete rest.

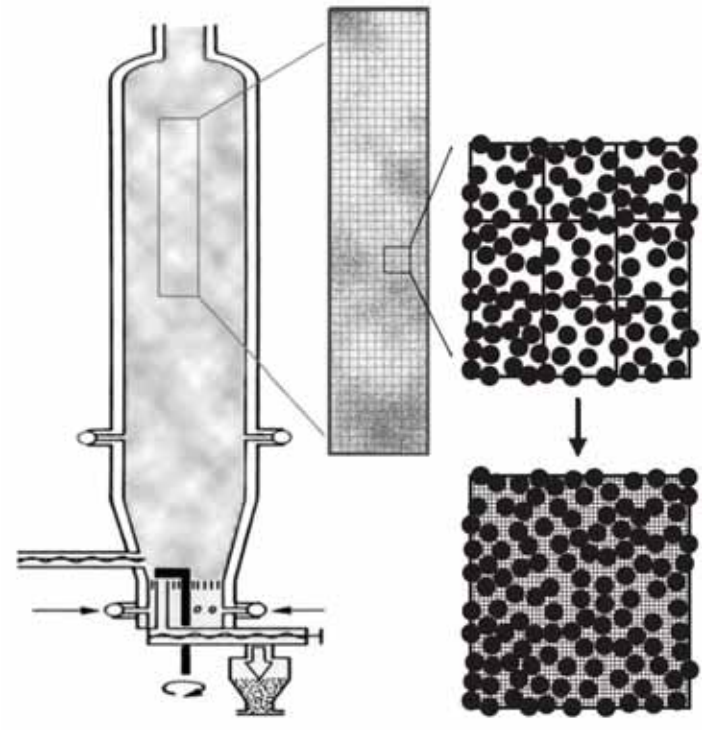

Figure 6.1: Example of macroscopic particle-based modelling. On the left is a fluidized bed on a life-size scale (grayscale indicates solid volume fraction). A section of the fluidized bed is modelled using a discrete particle method (top right), where the gas flow around the particles (black dots) is not resolved, but effective drag relations are used, treating the particles as point sources of momentum on the fluid. When using direct numerical simulations, the gas flow around the particles *is* fully resolved (bottom right), but this necessitates a grid size which is much smaller than the particle diameter. Adapted from [30].

Particle-based simulations on the granular scale include discrete element methods (DEM), direct numerical simulations (DNS), and discrete particle methods (DPM).[1]

The **discrete element method** is used when the effects of the surrounding medium (gas or liquid) are negligible, either because the gravity and inertial forces are orders of magnitude larger than the hydrodynamic drag forces (e.g. large particles moving relatively slowly through air) or because the particles are at such high solid volume fraction that the interactions between the particles dominate the hydrodynamic forces (e.g. particles packed together and moving slowly through a hopper).

The term **direct numerical simulation** is used when the particles *do* feel the surrounding fluid medium and the fluid flow around the particles is fully resolved. DNS is the most precise method of dealing with the fluid flow, by solving the Navier-Stokes equations using a grid size much smaller than the particle diameter (Figure 6.1).

In some cases – particularly for particles in a gas – it is possible to find accurate effective drag relations and treat the particles as point sources of momentum on the fluid. The fluid flow around the particles is *unresolved*, allowing us to use pre-averaged Navier-Stokes equations and a grid size much larger than the particle diameter, see Figure 6.1. We refer to such unresolved methods as **discrete particle methods**.

Let us start by defining the equations of motion for the particles.

[1]There exist also methods in which the solid particle phase is treated as a continuum. When the particles are embedded in a fluid, such methods are referred to as two-fluid models (TFM) because the particles and fluid are treated as two interpenetrating continua. An important ingredient of such models is an effective description of the particle stress.

6.3 Equations of motion for the particles in a granular system

The equations of motion for the particles in a DEM/DNS/DPM model are very similar to those of molecular dynamics simulations. Newton's equations of motion of individual particles are integrated using forces on the particles. What is different is the origin of these forces: they do not only contain conservative elements (which depend on the particle positions), but also dissipative particle-particle elements (which depend on the particle positions *and* velocities) and possibly fluid-induced forces (which depend on the pressure and velocity fields of the surrounding fluid). Moreover, particle *rotations* are usually taken explicitly into account requiring specification of the torques on particles.

Rigid body rotations

When rotations are taken into account, one usually assumes that the granular particle translate and rotate at rigid bodies. The equations of motion for rigid body motion are:

$$m_i \frac{\mathrm{d}\mathbf{v}_i}{\mathrm{d}t} = \mathbf{F}_i \tag{6.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{I}_i \cdot \omega_i) = \mathbf{T}_i \tag{6.2}$$

The first equation may look quite familiar by now. It describes the change in velocity $\mathbf{v}_i$ of the *centre of mass* of particle i due to the total force $\mathbf{F}_i$. The second equation is more complex. It describes the effect of the total torque $\mathbf{T}_i$ acting on particle i on the change in its angular velocity ω_i. The complexity resides in the moment of inertia tensor $\mathbf{I}_i$, which depends on the shape and current orientation of the particle.[2] In other words, we cannot simply integrate cartesian elements of the torque to update the angular velocity of the particle. Rather, we should solve the so-called Euler equations of motion for rigid body rotation.

Fortunately, for spherical particles the inertia tensor is diagonal and independent of its orientation. For a sphere of mass m_i and radius R_i we have:

$$\mathbf{I}_i = \begin{bmatrix} \frac{2}{5}m_i R_i^2 & 0 & 0 \\ 0 & \frac{2}{5}m_i R_i^2 & 0 \\ 0 & 0 & \frac{2}{5}m_i R_i^2 \end{bmatrix}, \qquad \text{(spheres)} \tag{6.3}$$

meaning that we can rewrite Eq. (6.2) as

$$I_i \frac{\mathrm{d}\omega_i}{\mathrm{d}t} = \mathbf{T}_i, \qquad \text{(spheres)} \tag{6.4}$$

[2] The term $\mathbf{I}_i \cdot \omega_i$ is also known as the angular momentum $\mathbf{L}_i$ of the particle. The equation of motion for rotations, $\mathrm{d}\mathbf{L}_i/\mathrm{d}t = \mathbf{T}_i$, is completely analogous to Newton's equation for translations in terms of momenta $\mathbf{p}_i = m_i\mathbf{v}_i$, namely $\mathrm{d}\mathbf{p}_i/\mathrm{d}t = \mathbf{F}_i$.

with $I_i = \frac{2}{5} m_i R_i^2$ the moment of inertia of particle (sphere) i. Numerically solving the angular velocity of rigid spherical particles is therefore as simple as solving the translational velocity. An equivalent of the velocity Verlet algorithm[3] for rigid spheres could be programmed as follows:

$$\mathbf{v}_i(t+\Delta t/2) = \mathbf{v}_i(t) + \frac{1}{2}\frac{\mathbf{F}_i(t)}{m_i}\Delta t, \tag{6.5}$$

$$\omega_i(t+\Delta t/2) = \omega_i(t) + \frac{1}{2}\frac{\mathbf{T}_i(t)}{I_i}\Delta t, \tag{6.6}$$

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \mathbf{v}_i(t+\Delta t/2)\Delta t, \tag{6.7}$$

(then evaluate force and torque at $t+\Delta t$)

$$\mathbf{v}_i(t+\Delta t) = \mathbf{v}_i(t+\Delta t/2) + \frac{1}{2}\frac{\mathbf{F}_i(t+\Delta t)}{m_i}\Delta t, \tag{6.8}$$

$$\omega_i(t+\Delta t) = \omega_i(t+\Delta t/2) + \frac{1}{2}\frac{\mathbf{T}_i(t+\Delta t)}{I_i}\Delta t. \tag{6.9}$$

Note that the force and torque need to be evaluated only once per time step Δt because the new force (torque) in the current step will be the old force (torque) in the next step.

Forces and torques in DEM, DNS and DPM simulations

So what are the forces and torques on a granular particle? Generally, we may write the force $\mathbf{F}_i$ and torque $\mathbf{T}_i$ on particle i with volume V_i as:

$$\mathbf{F}_i = \mathbf{F}_{fluid,i} + m_i\mathbf{g} + \mathbf{F}_{contact,i} + \mathbf{F}_{pp,i}. \tag{6.10}$$

$$\mathbf{T}_i = \mathbf{T}_{fluid,i} + \mathbf{T}_{contact,i} + \mathbf{T}_{pp,i}. \tag{6.11}$$

These forces and torques are, respectively [17]:

- The fluid-induced (hydrodynamic) forces and torques on the particle.
- A gravity force.
- Contact forces and torques due to direct collisions with neighbouring particles.
- Longer range particle-particle forces and torques such as due to Van der Waals and electrostatic interactions.

The difference between DEM, DNS and DPM models lies in the treatment of the fluid-induced forces. In DEM the fluid-induced forces are neglected. In DNS the fluid is fully resolved on a grid finer than the particles, whereas in DPM the fluid is unresolved and particles are treated as point sources of momentum, using effective drag force correlations.

For all models we need the contact forces. We will deal with this in section 6.4. Then we will deal with fluid-induced forces in section 6.5. We will not explicitly deal

[3] Higher order schemes such as Beeman and Runge-Kutta methods may also be used.

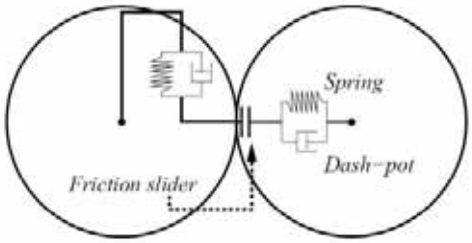

Figure 6.2: Graphical representation of the linear spring-dashpot soft sphere model for granular particles. After Hoomans [32].

with longer range particle-particle forces because they are very similar to the longer range forces occurring in molecular dynamics and mesoscale simulations.

6.4 Dissipative collisions: contact models

In the field of granular flow, a model specifying the forces and torques on the particles due to direct collisions is usually referred to as a **contact model**. Usually the contact force and torque on particle i are approximated as pair sums, i.e.

$$\mathbf{F}_{contact,i} = \sum_{j \neq i} \mathbf{F}_{ij}, \tag{6.12}$$

$$\mathbf{T}_{contact,i} = \sum_{j \neq i} \mathbf{T}_{ij}, \tag{6.13}$$

where $\mathbf{F}_{ij}$ is defined as the contact force on i due to its contact with particle j, and similarly for the torque. Of course, because contact forces are short ranged, we can use a cell linked list and/or a neighbourlist to efficiently deal with these pair sums.

Because of Newton's third law (action equals minus reaction, or $\mathbf{F}_{ij} = -\mathbf{F}_{ji}$) the total momentum of two particles in contact is conserved.[4] The energy, however, is usually not conserved. More precisely, energy is transferred to more microscopic degrees of freedom during during a collision between two granular particles. An important ingredient of a contact models is therefore the inclusion of dissipative forces.

Linear spring-dashpot soft sphere model

One of the simplest and most used contact models is the linear spring-dashpot soft sphere model [16], depicted schematically in Figure 6.2.[5] The total contact force be-

[4] And similarly the *total* angular momentum, i.e. angular momentum of the particles around their respective centres of mass plus angular momentum associated with the trajectories of the centres of mass, is conserved.

[5] The other much used contact model is the hard sphere model. Similar to the soft sphere model, the hard sphere model is characterised by a normal and tangential coefficient of restitution and a coefficient of friction. The positions of such hard spheres are not propagated using constant time steps, but rather by calculating the next collision event. We do not treat such event-driven simulations in these lectures. Hard sphere models are unfit for simulations at high densities because of divergent collision frequencies.

tween two spherical particles i and j of radius R_i and R_j, respectively, is divided into a normal force $\mathbf{F}_{ij,n}$ and a tangential force $\mathbf{F}_{ij,t}$, i.e. $\mathbf{F}_{ij} = \mathbf{F}_{ij,n} + \mathbf{F}_{ij,t}$.

Normal contact force

The normal contact force on particle i due to its contact with particle j is calculated according to

$$\mathbf{F}_{ij,n} = k_n \delta_n \hat{\mathbf{n}}_{ij} - \eta_n \mathbf{v}_{ij,n}, \tag{6.14}$$

The first part represents a conservative force, where k_n is the spring stiffness (in units N/m) in the normal direction, and δ_n is the overlap (in units m) between the particles in the normal direction:

$$\delta_n = (R_i + R_j) - \left|\mathbf{r}_i - \mathbf{r}_j\right|. \tag{6.15}$$

The normal direction is defined as the unit vector pointing from the centre of j to the centre of i:

$$\hat{\mathbf{n}}_{ij} = \frac{\mathbf{r}_i - \mathbf{r}_j}{\left|\mathbf{r}_i - \mathbf{r}_j\right|}. \tag{6.16}$$

The dissipative force in the normal direction is controlled by η_n, the damping coefficient (in units kg/s)) in the normal direction, and $\mathbf{v}_{ij,n}$, the normal component (in the normal direction) of the relative velocity at the point of contact.

The relative velocity may need some explanation. Figure 6.3 shows two granular particles with centre of mass velocities $\mathbf{v}_i$ and $\mathbf{v}_j$ and angular velocities (around their respective centres of mass) ω_i and ω_j. In practice the normal spring stiffness is chosen so large that the maximum overlap is of the order of a percent of the particle diameter. Therefore, the relative velocity at the point of contact can be approximated by the difference between the local surface velocity of particle i, $\mathbf{v}_i^{loc} = \mathbf{v}_i - R_i \hat{\mathbf{n}}_{ij} \times \omega_i$, and the local surface velocity of particle j, $\mathbf{v}_j^{loc} = \mathbf{v}_j + R_j \hat{\mathbf{n}}_{ij} \times \omega_j$. In other words,

$$\mathbf{v}_{ij} = \left(\mathbf{v}_i - \mathbf{v}_j\right) - \left(R_i \omega_i + R_j \omega_j\right) \times \hat{\mathbf{n}}_{ij} \tag{6.17}$$

The normal component in the normal direction of the relative velocity is thus defined as

$$\mathbf{v}_{ij,n} = \left(\mathbf{v}_{ij} \cdot \hat{\mathbf{n}}_{ij}\right) \hat{\mathbf{n}}_{ij}. \tag{6.18}$$

In the following section we will need the tangential velocity, which is the remaining part:

$$\mathbf{v}_{ij,t} = \mathbf{v}_{ij} - \mathbf{v}_{ij,n}. \tag{6.19}$$

Check for yourself that $\mathbf{v}_{ij,t}$ is indeed always tangential to the surfaces of the particles at the contact point.

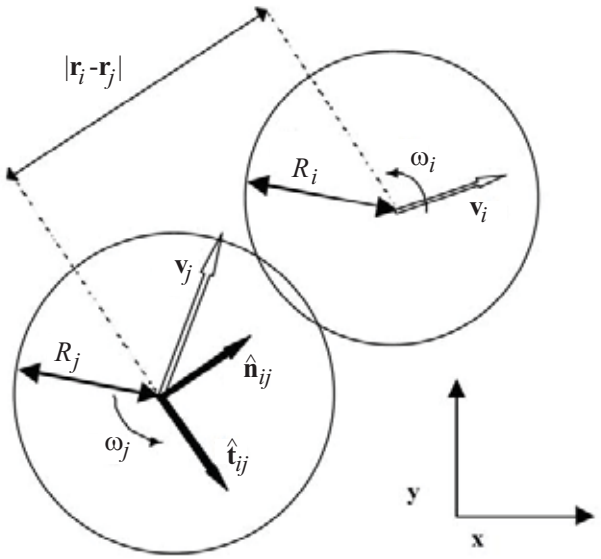

Figure 6.3: Coordinate system for two granular particles i and j of radius R_i and R_j, with the definition of the unit normal vector $\hat{\mathbf{n}}_{ij}$ and tangential normal vector $\hat{\mathbf{t}}_{ij}$.

Tangential contact force: sticking and sliding

For the tangential contact force on partice i due to its contact with particle j, two cases should be distinguished, namely "sticking" and "sliding" contacts [32, 39]. If the tangential velocity is sufficiently high, the impact can be described as sliding during the entire collisions. However, when after the initial sliding phase the relative tangential velocity between the two colliding particles becomes zero, the impact of the particles belongs to the sticking case.

In the sticking case, the tangential contact force has a form similar to the normal force:

$$\mathbf{F}_{ij,t} = -k_t \delta_t \hat{\mathbf{t}}_{ij} - \eta_t \mathbf{v}_{ij,t}. \tag{6.20}$$

Here k_t is the spring stiffness (in units N/m) in the tangential direction $\hat{\mathbf{t}}_{ij}$, which is defined as the unit vector along the direction of the tangential velocity,

$$\hat{\mathbf{t}}_{ij} = \frac{\mathbf{v}_{ij,t}}{|\mathbf{v}_{ij,t}|}. \tag{6.21}$$

The overlap in the tangential direction δ_t (in units m) is not a trivial quantity, but is defined as the length of the time integral of the tangential velocity since the start of the contact at t_0,

$$\delta_t = \left| \int_{t_0}^{t} \mathbf{v}_{ij,t}(t') \mathrm{d}t' \right|. \tag{6.22}$$

The damping coefficient in the tangential direction η_t (in kg/s) plays a similar role to that in the normal direction.

The above tangential force applies to the sticking case. But what about sliding? We already encountered a sliding case in section 2.3 when we considered dissipation of energy for a block sliding down an inclined plane. We considered the empirical fact that when sliding occurs, the friction force (acting tangential to the surface) scales linearly with the normal force $\left|\mathbf{F}_{ij,n}\right|$. The coefficient of proportionality is called the coefficient of (Coulomb or kinetic) friction μ_f. So the coefficient of friction poses an *upper limit* to the tangential force that two particles can exert on each other. In formula:

$$\mathbf{F}_{ij,t} = \begin{cases} \text{Eq. (6.20)} & \text{if } \left|\mathbf{F}_{ij,t}\right| \leq \mu_f \left|\mathbf{F}_{ij,n}\right| \quad \text{(sticking)} \\ -\mu_f \left|\mathbf{F}_{ij,n}\right| \hat{\mathbf{t}}_{ij} & \text{if } \left|\mathbf{F}_{ij,t}\right| > \mu_f \left|\mathbf{F}_{ij,n}\right| \quad \text{(sliding)} \end{cases} \tag{6.23}$$

The tangential force leads to a torque on each of the particles i and j. The torque on particle i due to its contact with particle j is given by

$$\mathbf{T}_{ij} = -R_i \hat{\mathbf{n}}_{ij} \times \mathbf{F}_{ij,t}. \tag{6.24}$$

The minus sign arises because $\hat{\mathbf{n}}_{ij}$ points from the contact point to the centre of mass of i. Using the same reasoning, the torque on j due to its contact with i is given by $\mathbf{T}_{ji} = R_j \hat{\mathbf{n}}_{ij} \times (-\mathbf{F}_{ij,t}) = -R_j \hat{\mathbf{n}}_{ij} \times \mathbf{F}_{ij,t}$.

Contact parameters

To determine the normal and tangential forces and torques between the particles we need to know five parameters (assuming all particles are equal): the normal and tangential spring stiffness k_n and k_t, the normal and tangential damping coefficients η_n and η_t, and the friction coefficient μ_f.

Experimentally, it is very difficult to measure the damping coefficients. Rather, in experiments we can measure the coefficients of restitution e_n and e_t in the normal and tangential directions. The normal coefficient of restitution is defined as the ratio of the normal velocity at the end of an impact and the normal velocity at the start of an impact, and similarly for the tangential coefficient of restitution:

$$e_n \equiv -\frac{\mathbf{v}_{ij,n,end}}{\mathbf{v}_{ij,n,start}}, \tag{6.25}$$

$$e_t \equiv \frac{\mathbf{v}_{ij,t,end}}{\mathbf{v}_{ij,t,start}}. \tag{6.26}$$

The minus sign in the definition of e_n arises because the particle reverses its direction during impact. For a perfectly elastic collision we have $e_n = e_t = 1$. Real macroscopic particles never collide perfectly elastically. For example, for millimetre sized glass spheres we have $e_n \approx 0.93$.

The higher the amount of dissipation (damping) during an impact, the lower (further away from 1) the coefficient of restitution. We therefore expect that there exists a relation between the damping coefficients and the coefficients of restitution. Indeed,

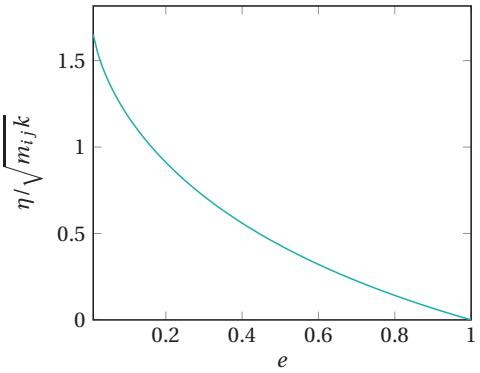

Figure 6.4: Relation between the damping coefficient η and coefficient of restitution e. For the normal direction the vertical axis is scaled by $\sqrt{m_{ij}k_n}$, for the tangential direction by $\sqrt{m'_{ij}k_t}$ (see main text).

since the equations of motion for the collision process of a pair of particle are essentially those of a damped harmonic oscillator, it is possible to solve the equations analytically (not shown here). The analytical equations predicts the following relations between η_n and e_n and between η_t and e_t:

$$\eta_n = \frac{-2\ln e_n \sqrt{m_{ij}k_n}}{\sqrt{\pi^2+\ln^2 e_n}} \tag{6.27}$$

$$\eta_t = \frac{-2\ln e_t \sqrt{m'_{ij}k_t}}{\sqrt{\pi^2+\ln^2 e_t}} \tag{6.28}$$

where $m_{ij} = m_i m_j/(m_i+m_j)$ is the reduced mass of particles i and j, and $m'_{ij} = \frac{2}{7}m_{ij}$ (see also Figure 6.4). If all particles have the same mass m, then $m_{ij} = m/2$, while for a particle-wall collision we may treat the wall as a particle of infinite mass, making the reduced mass equal to the mass of the free particle. The analytical solution also gives us the contact time for normal and tangential directions:

$$t_{contact,n} = \sqrt{\frac{m_{ij}(\pi^2+\ln^2 e_n)}{k_n}} \tag{6.29}$$

$$t_{contact,t} = \sqrt{\frac{m'_{ij}(\pi^2+\ln^2 e_t)}{k_t}}. \tag{6.30}$$

This last result is very important. Just think about it: how could the contact time in the normal direction be different from the contact time in the tangential direction? For a *consistent* contact model the contact times in the normal and tangential directions should be exactly equal.[6] This extra requirement reduces the number of free parame-

[6]This is also important for a proper energy balance. For example, if the normal collision is finished before the tangential collision is finished, the particles no longer overlap, but there is still some energy stored in the tangential spring. This energy will then be lost, resulting in an effectively lower coefficient of restitution than intended.

ters by one. Usually the tangential spring stiffness is expressed in the other parameters:

$$\frac{k_t}{k_n} = \frac{2}{7}\frac{\pi^2 + \ln^2 e_t}{\pi^2 + \ln^2 e_n}. \tag{6.31}$$

Although the normal spring stiffness can be determined from the Youngs modulus and radius of the particle, it usually yields a very high value, which implies the use of a very small time step, which is undesirable from a computational point of view [17]. In practice k_n is set to lower values, while ensuring that the normal overlap is kept small, i.e. typically below 1% of the particle diameter.

6.5 Coupling to a fluid flow

The influence of a surrounding fluid on the dynamics of granular particles may be ignored if we are studying dense particle packings or if we are dealing with large particles moving relatively slowly through a dilute gas such as air. In these special cases a contact model, such as the spring-dashpot model discussed in the previous section, together with gravity and possibly long-range particle-particle forces, suffices to make predictions about the system. This is the field of discrete element method (DEM) simulations.

In most other cases, however, the liquid or gas will influence the granular particle trajectories. We then enter the fields of direct numerical simulations (DNS) and discrete particle method (DPM) simulations.

A brief digression into direct numerical simulations (DNS)

The case of solid particles flowing through a liquid is arguably more complex than that of solid particles flowing through gas, because both long range hydrodynamic interactions and short range lubrication forces are more dominant in liquid flows. Actually the same problems are encountered in simulations of macroscopic solid-liquid flows as in simulations of mesoscale colloidal suspensions: because of the long-range nature of hydrodynamic interactions, the dynamics of a solid particle depends on the positions and velocities of a very large number of other particles. Although one of the complications of mesoscale simulations is avoided – Brownian motion is absent in macroscopic particles – we get in return another complication, namely inertia of the particles and fluid. High Reynolds number hydrodynamic flow is practically impossible to predict analytically.

The solution is to perform detailed simulations of particles embedded in a fluid governed by the full Navier-Stokes equations. To ensure accurate predictions of all hydrodynamic forces, the Navier-Stokes partial differential equations must be solved numerically on a grid which is much *smaller* than the particle diameter. These types

of simulations are collectively referred to as **direct numerical simulations** (DNS).[7] In DNS the coupling with the embedded solid particles is accomplished by enforcing no-slip boundary conditions between the liquid and the solid particles at the surface of the particles. The fluid-induced forces and torques on the particles follow from the pressure and stress tensor of the fluid at the boundary of the volume of particle i, which we denote symbollically as ∂V_i:

$$\mathbf{F}_{fluid,i} = -\int_{\partial V_i} P(a)\hat{\mathbf{n}}(a)\mathrm{d}a + \int_{\partial V_i} \bar{\mathbf{S}}(a)\cdot\hat{\mathbf{n}}(a)\mathrm{d}a, \tag{6.32}$$

$$\mathbf{T}_{fluid,i} = -\int_{\partial V_i} P(a)\left(\mathbf{r}(a)-\mathbf{r}_i\right)\times\hat{\mathbf{n}}(a)\mathrm{d}a + \int_{\partial V_i} \left(\mathbf{r}(a)-\mathbf{r}_i\right)\times\left(\bar{\mathbf{S}}(a)\cdot\hat{\mathbf{n}}(a)\right)\mathrm{d}a. \tag{6.33}$$

Here $\int_{\partial V_i}\ldots\mathrm{d}a$ indicates a surface integral over the boundary of particle i. $\hat{\mathbf{n}}(a)$ is the normal unit vector pointing away from the particle at a particular location $\mathbf{r}(a)$ of its surface. In the above expressions $\bar{\mathbf{S}}$ is the fluid-phase stress tensor, given by

$$\bar{\mathbf{S}} = -\left(\lambda-\frac{2}{3}\mu\right)(\boldsymbol{\nabla}\cdot\mathbf{u})\,\bar{\mathbf{I}} - \mu\left(\boldsymbol{\nabla}\mathbf{u}+(\boldsymbol{\nabla}\mathbf{u})^T\right), \tag{6.34}$$

where the bulk viscosity λ is usually set to zero for gases. An explicit example of this type of boundary integral calculation for the case of (zero Reynolds number) Stokes flow of a sphere can be found in Appendix A.2. Note that for a sphere the pressure cannot exert a torque on a particle because the normal unit vector $\hat{\mathbf{n}}(a)$ is always parallel to $(\mathbf{r}(a)-\mathbf{r}_i)$.

In these lectures we will not further delve into the details of DNS simulations, as they are the topic of other (Computation Fluid Dynamics) courses.

Discrete particle model (DPM) simulations

When solid particles are flowing through a gas, the situation is relatively simpler (though still quite complex!) because usually the lubrication forces are not as dominant as in a liquid – the viscosity of a gas is much lower than in a liquid. Although multiple particles in a gas flow affect each other, direct numerical simulations of solid particles in gas flows have shown that the multi-particle effect on the drag force experienced by a solid particle may be approximated as a dependence of the drag on the local solid volume fraction ϕ only. In other words, the drag forces are influenced by the presence of other solid particles, but experience has shown that their influence may be approximated as only indirectly, through a dependence of the drag forces on the local solid volume fraction. Making use of this fact, we may reverse the situation and solve the Navier-Stokes equations on a grid which is much *larger* than the size of the particles. This is exactly what is done in discrete particle model (DPM) simulations. DPM simulations

[7] Sometimes lattice Boltzmann simulations (without thermal fluctuations) are employed as an efficient Navier-Stokes solver. Such simulations are then also referred to as direct numerical simulations.

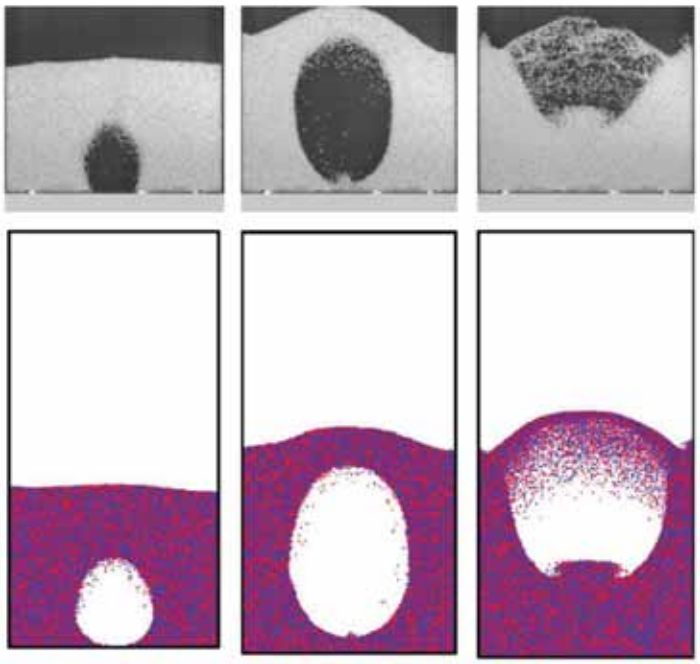

Figure 6.5: Injection of a single bubble in the centre of a fluidized bed (bed width: 0.30 m), containing spherical glass beads of 2.5 mm diameter at incipient fluidisation conditions. Comparison of experimental data (top) with discrete particle model (DPM) simulations (bottom) for 0.1, 0.2 and 0.4 s after bubble injection [12].

have been applied with succes to, e.g., the flow of particles in fluidized bed reactors (Figure 6.5).

Gas-induced forces on a particle

In DPM simulations the fluid-induced force on particle i of volume V_i consists of two terms:

$$\mathbf{F}_{fluid,i} = -V_i \nabla P + \frac{V_i \beta}{\phi} (\mathbf{u} - \mathbf{v}_i) . \tag{6.35}$$

The first term is the buoyant force on particle i, driven by pressure gradients in the surrounding fluid. The second term is the drag force due to a difference between the local gas velocity $\mathbf{u}$ and the particle velocity $\mathbf{v}_i$; β is the so-called inter-phase momentum transfer coefficient (in units of kg/(s.m^3)) and ϕ the local solid volume fraction. Regarding the torque $\mathbf{T}_i$ on particle i, the surrounding medium could exert a torque (think of a fluid with a high amount of vorticity), but in practice this is often neglected in DPM models.[8]

To calculate the fluid-induced force we therefore need *local* estimates of the solid volume fraction ϕ and the inter-phase momentum transfer coefficient β.

Usually the volume fraction is stored at discrete grid locations of the computational cells for the fluid. When the volume of the smallest computational cell for the fluid is

[8] For spheres in a gas (such as in fluidized beds) neglecting the gas-induced torque is acceptable, but for elongated particles or for particles suspended in a liquid this is most probably not allowed.

much larger than the volume of a particle, the mapping of properties from the (Eulerian) computational grid to the (Lagrangian) particle positions (and vice versa) can be done in a straightforward manner through volume-weighing techniques [32] and [18]. The idea is to calculate the local solid volume fraction ϕ_{cell} of a cell as

$$\phi_{cell} = \frac{1}{V_{cell}} \sum_{i \in cell} f_{cell}^{i} V_i, \tag{6.36}$$

where V_{cell} is the volume of the cell and f_{cell}^{i} is the fractional volume of particle i residing in the cell under consideration. After obtaining the solid volume fractions of all cells, the local solid volume fraction ϕ at the location of a specific particle is obtained by trilinear interpolation.

The inter-phase momentum transfer coefficient is frequently modelled by combining the Ergun equation [22] for dense regimes ($\phi > 0.2$),

$$F_{\text{Ergun}} = \frac{\beta d^2}{\mu} = 150 \frac{\phi^2}{1-\phi} + 1.75 \phi \text{Re}, \tag{6.37}$$

and the correlation proposed by Wen and Yu [61] for the more dilute regimes ($\phi < 0.2$),

$$F_{\text{Wen\&Yu}} = \frac{\beta d^2}{\mu} = \frac{3}{4} C_D \text{Re} \phi \left(1-\phi\right)^{-2.65}, \tag{6.38}$$

$$C_D = \begin{cases} 24\left(1+0.15\text{Re}^{0.687}\right)/\text{Re} & \text{if Re} < 10^3, \\ 0.44 & \text{if Re} > 10^3, \end{cases} \tag{6.39}$$

where d is the particle diameter and μ is the gas viscosity. Re is the particle Reynolds number, defined as

$$\text{Re} = \frac{(1-\phi)\rho_g |\mathbf{u} - \mathbf{v}_i| d}{\mu}, \tag{6.40}$$

where ρ_g is the gas density. The particle Reynolds number is usually much larger than unity, which gives rise to an unrealistic jump in the drag curve at $\phi = 0.2$ [17]. This problem can be circumvented by using the least value of Eqs. (6.37) and (6.38) for the calculation of β.

Many other drag relations exist, of which we mention only one, by Beetstra et al. [8],

$$F_{\text{Beetstra}} = \frac{\beta d^2}{\mu} = A \frac{\phi^2}{1-\phi} + B\phi\text{Re}, \tag{6.41}$$

$$A = 180 + \frac{18\left(1-\phi\right)^4}{\phi}\left(1+1.5\sqrt{\phi}\right), \tag{6.42}$$

$$B = \frac{0.31\left[\left(1-\phi\right)^{-1} + 3\phi\left(1-\phi\right) + 8.4\text{Re}^{-0.343}\right]}{1+10^{3\phi}\text{Re}^{-0.5-2\phi}}. \tag{6.43}$$

This drag relation is frequently used in our DPM simulations of gas-fluidised beds.

Particle-induced forces on gas

The gas does not only exert a force on the particles, but by Newton's third law the particles also exert a force on the gas. At very low solid volume fractions ($\phi < 10^{-4}$) this so-called two-way coupling may be neglected (leading to a one-way coupled system), but in general it may not. The gas-phase hydrodynamics are then calculated from the volume-averaged Navier-Stokes equations [17]:

$$\frac{\partial}{\partial t}\left((1-\phi)\rho_g\right) + \nabla\cdot\left((1-\phi)\rho_g\mathbf{u}\right) = 0 \tag{6.44}$$

$$\frac{\partial}{\partial t}\left((1-\phi)\rho_g\mathbf{u}\right) + \nabla\cdot\left((1-\phi)\rho_g\mathbf{uu}\right) = -(1-\phi)\nabla P - \nabla\cdot\left((1-\phi)\bar{\mathbf{S}}\right) - \mathbf{S}_p + (1-\phi)\rho_g\mathbf{g}. \tag{6.45}$$

The gas-phase stress tensor $\bar{\mathbf{S}}$ is given by Eq. (6.34). The two-way coupling between the gas phase and the particles is enforced via the sink term $\mathbf{S}_p$ in the momentum equation of the gas-phase, which is computed from [17]:

$$\mathbf{S}_p = \frac{1}{V_{cell}}\int_{V_{cell}}\sum_{i=0}^{N}\frac{V_i\beta}{\phi}(\mathbf{u}-\mathbf{v}_i)\,D(\mathbf{r}-\mathbf{r}_i)\mathrm{d}V. \tag{6.46}$$

Here the sum is over all N particles inside the system. The distribution function D distributes the reaction force acting on the gas phase to the (Eulerian) gas grid. When the volume of the smallest computational cell for the fluid is much larger than the volume of a particle, the mapping of properties from the (Lagrangian) particle positions to the (Eulerian) computational grid and vice versa can be done in a straightforward manner through trilinear interpolation.

6.6 Stochastic methods to model the dynamics of dilute granular flow

Note that all the above methods (DEM, DNS and DPM) are deterministic methods: collisions between the particles are detected based on precise rules for overlap. In certain limiting cases it is possible to use stochastic methods to deal with the collisions. Particularly, when we are dealing with relatively dilute streams of particles, sufficiently perturbed by gas flow to experience random velocity fluctuations, we may take advantage of the fact that collisions are rare on a particle-to-particle basis (large mean-free-paths), yet common enough to apply probabilistic collision rules.

For example, when modelling liquid droplets in the jet emerging from a spray dryer, we are dealing with hundreds of millions or even billions of particles (droplets). It is computationally too expensive to deal with such particles in a deterministic fashion. Rather, we can use the direct simulation Monte Carlo (DSMC) technique [10], briefly touched upon in chapter 5, to estimate the *probability* that a given particle i of size d_i moving with velocity $\mathbf{v}_i$ will collide with another nearby particle j of size d_j moving

with velocity $\mathbf{v}_j$.[9] We then let random pairs of particles (within a prescribed range) collide with each other with the correct probability. Further speed-up may be achieved by not actually simulating each and every particle but by tracking representative droplets, each of which represents a large number (say 10^2 to 10^4) real particles.

Of course such a stochastic approach is only allowed as long as many random collisions occur between the particles (in relatively dilute streams) and we are not interested in the exact deterministic path of individual particles.

6.7 Limitations of macroscopic particulate models

A number of methods exist to deal with the generally high Reynolds number dynamics of granular particles. Let us summarize the range of applicability and limitations of each method.

- **Discrete element model** (DEM) simulations neglect all hydrodynamic effects of the fluid surrounding the granular particles. The advantage is that this makes DEM fast compared to methods that do take into account the flow of liquid or gas; systems containing a million particles are not uncommon. Disadvantage is that the method is limited to those cases where we can neglect the fluid hydrodynamics. Examples include large particles moving slowly through a dilute gas and slow flow of dense packings of particles.

- In **direct numerical simulations** (DNS) the effects of the fluid are included in great detail. Because the grid cell size – used to solve the discretised Navier-Stokes equations – is much smaller than the particle size, DNS simulations are usually limited to very small system sizes (approximately 100 particles).

- In **discrete particle model** (DPM) simulations we use effective drag relations – these may be obtained from experiments or DNS simulations – to model the coupling with the fluid phase. Because the grid cell size is much larger than the particle size, DPM simulations can deal with much larger systems than DNS simulations (typically a few times 10^5 particles). A limitation is that in systems where the hydrodynamic forces are important, the accuracy of DPM simulations depends critically on the applicability and accuracy of the effective drag relations.

- In stochastic methods such as **direct simulation Monte Carlo** (DSMC) the collisions between particles are not detected deterministically, but rather stochastically, based on collision probabilities. Also many particles may be represented by just a single representative particle. Advantage is a huge speed-up, allowing for simulations of 10^6 to 10^9 particles. However, these methods are usually limited to relatively dilute particle flows where many random collisions occur between the particles.

[9] Using kinetic theorey, originally developed for dilute gases, it is possible to show that this probability will grow with the collision cross-section area $(d_1 + d_2)^2$ and with the relative velocity $|\mathbf{v}_i - \mathbf{v}_j|$.

APPENDIX A

HYDRODYNAMIC FORCES ON SLOWLY MOVING SPHERES

In this appendix we will calculate hydrodynamic (friction) forces on slowly moving spheres (Reynolds number $\ll 1$). We will first introduce the Navier-Stokes and Stokes equations, then deal with a single sphere, and finally deal with hydrodynamic interactions in a suspension of many spheres.

A.1 Navier-Stokes and Stokes equations

To formulate the basic equations for the fluid we utilize the conservation of mass and momentum. The conservation of mass is expressed by the continuity equation

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} = -\rho \boldsymbol{\nabla} \cdot \mathbf{v}, \tag{A.1}$$

and the conservation of momentum by the Navier-Stokes equation

$$\rho \frac{\mathrm{D}}{\mathrm{D}t}\mathbf{v} = \boldsymbol{\nabla} \cdot \bar{\mathbf{S}}. \tag{A.2}$$

Here $\rho(\mathbf{r}, t)$ is the fluid density, $\mathbf{v}(\mathbf{r}, t)$ the fluid velocity, $\mathrm{D}/\mathrm{D}t \equiv \mathbf{v} \cdot \boldsymbol{\nabla} + \partial/\partial t$ the total derivative, and $\bar{\mathbf{S}}$ is the stress tensor.

We now have to specify the nature of the stress tensor $\bar{\mathbf{S}}$. For a viscous fluid, friction occurs when the distance between two neighbouring fluid elements changes, i.e. they move relative to each other. Most simple fluids can be described by a stress tensor which consists of a part which is independent of the velocity, and a part which depends linearly on the derivatives $\partial v_\alpha / \partial r_\beta$, i.e., where the friction force is proportional to the

instantaneous relative velocity of the two fluid elements.[1] The most general form of the stress tensor for such a fluid is

$$S_{\alpha\beta} = \mu\left\{\frac{\partial v_\alpha}{\partial r_\beta} + \frac{\partial v_\beta}{\partial r_\alpha}\right\} - \left\{P + \left(\frac{2}{3}\mu - \kappa\right)\boldsymbol{\nabla}\cdot\mathbf{v}\right\}\delta_{\alpha\beta}, \tag{A.3}$$

where μ is the dynamic shear viscosity, κ the bulk viscosity, which is the resistance of the fluid against compression, and P the pressure.

Many flow fields of interest can be described assuming that the fluid is incompressible, i.e. that the density along the flow is constant. In that case $\boldsymbol{\nabla}\cdot\mathbf{v} = 0$, as follows from Eq. (A.1). Assuming moreover that the velocities are small, and that the second order non-linear term $\mathbf{v}\cdot\boldsymbol{\nabla}\mathbf{v}$ may be neglected, we obtain Stokes equation for incompressible flow

$$\rho\frac{\partial \mathbf{v}}{\partial t} = \mu\nabla^2\mathbf{v} - \boldsymbol{\nabla}P \tag{A.4}$$

$$\boldsymbol{\nabla}\cdot\mathbf{v} = 0. \tag{A.5}$$

This is the starting point for our calculation of the hydrodynamic forces on spheres.

A.2 Friction on a single slowly moving sphere

First consider a single sphere of radius a moving with velocity $\mathbf{v}_S$ in a quiescent liquid. Assume that the velocity field is stationary. Referring all coordinates and velocities to a frame which moves with velocity $\mathbf{v}_S$ relative to the fluid transforms the problem into one of a resting sphere in a fluid which, at large distances from the sphere, moves with constant velocity $\mathbf{v}_0 \equiv -\mathbf{v}_S$. The problem is best considered in spherical coordinates (see Fig. A.1),[2] $\mathbf{v}(\mathbf{r}) = v_r\hat{\mathbf{e}}_r + v_\theta\hat{\mathbf{e}}_\theta + v_\phi\hat{\mathbf{e}}_\phi$, so that $\theta = 0$ in the flow direction. By symmetry

[1] The calculations in this Appendix assume that the solvent is an isotropic, unstructured fluid, with a characteristic stress relaxation time which is much smaller than the time scale of any flow experiment. The stress response of such a so-called Newtonian fluid appears to be *instantaneous*. Newtonian fluids usually consist of small and roughly spherical molecules, e.g., water and light oils. Non-Newtonian fluids, on the other hand, usually consist of large or elongated molecules. Often they are structured, either spontaneously or under the influence of flow. Their characteristic stress relaxation time is experimentally accessible. As a consequence, the stress between two non-Newtonian fluid elements generally depends on the *history* of relative velocities, and contains an elastic part. Examples are polymers and self-assembling surfactants.

[2] In spherical coordinates the gradient, Laplacian and divergence are given by

$$\boldsymbol{\nabla}f = \hat{\mathbf{e}}_r\frac{\partial}{\partial r}f + \frac{1}{r}\hat{\mathbf{e}}_\theta\frac{\partial}{\partial\theta}f + \frac{1}{r\sin\theta}\hat{\mathbf{e}}_\phi\frac{\partial}{\partial\phi}f$$

$$\nabla^2 f = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2\frac{\partial}{\partial r}f\right) + \frac{1}{r^2\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\frac{\partial}{\partial\theta}f\right) + \frac{1}{r^2\sin^2\theta}\frac{\partial^2}{\partial\phi^2}f$$

$$\boldsymbol{\nabla}\cdot\mathbf{v} = \frac{1}{r^2}\frac{\partial}{\partial r}\left(r^2 v_r\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\theta}\left(\sin\theta\, v_\theta\right) + \frac{1}{r\sin\theta}\frac{\partial}{\partial\phi}v_\phi.$$

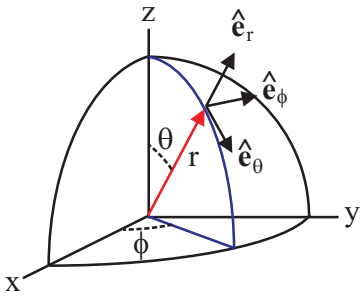

Figure A.1: Definition of spherical coordinates (r,θ,ϕ) and the unit vectors $\hat{\mathbf{e}}_r$, $\hat{\mathbf{e}}_\theta$, and $\hat{\mathbf{e}}_\phi$.

the azimuthal component of the fluid velocity is equal to zero, $v_\phi = 0$. The fluid flow at infinity gives the boundary conditions

$$\left.\begin{array}{rcl} v_r &=& v_0\cos\theta \\ v_\theta &=& -v_0\sin\theta \end{array}\right\} \text{ for } r\to\infty. \tag{A.6}$$

Moreover, we will assume that the fluid is at rest on the surface of the sphere (stick boundary conditions):

$$v_r = v_\theta = 0 \text{ for } r = a. \tag{A.7}$$

It can easily be verified that the solution of Eqs. (A.4) - (A.5) is

$$v_r = v_0\cos\theta\left(1-\frac{3a}{2r}+\frac{a^3}{2r^3}\right) \tag{A.8}$$

$$v_\theta = -v_0\sin\theta\left(1-\frac{3a}{4r}-\frac{a^3}{4r^3}\right) \tag{A.9}$$

$$p-p_0 = -\frac{3}{2}\frac{\mu v_0 a}{r^2}\cos\theta. \tag{A.10}$$

We shall now use this flow field to calculate the friction force exerted by the fluid on the sphere. The stress on the surface of the sphere results in the following force per unit area:

$$\begin{aligned}\mathbf{f} &= \bar{\mathbf{S}}\cdot\hat{\mathbf{e}}_r = \hat{\mathbf{e}}_r S_{rr} + \hat{\mathbf{e}}_\theta S_{\theta r} = -\hat{\mathbf{e}}_r\, p\big|_{(r=a)} + \hat{\mathbf{e}}_\theta \mu \left.\frac{\partial v_\theta}{\partial r}\right|_{(r=a)} \\ &= \left(-p_0+\frac{3\mu v_0}{2a}\cos\theta\right)\hat{\mathbf{e}}_r - \frac{3\mu v_0}{2a}\sin\theta\hat{\mathbf{e}}_\theta.\end{aligned} \tag{A.11}$$

Integrating over the whole surface of the sphere, only the component in the flow direction survives:

$$F = \int \mathrm{d}\Omega\, a^2\left[\left(-p_0+\frac{3\mu v_0}{2a}\cos\theta\right)\cos\theta + \frac{3\mu v_0}{2a}\sin^2\theta\right] = 6\pi\mu a v_0. \tag{A.12}$$

Transforming back to the frame in which the sphere is moving with velocity $\mathbf{v}_S = -\mathbf{v}_0$ through a quiescent liquid, we find for the fluid flow field

$$\mathbf{v}(\mathbf{r}) = \mathbf{v}_S\frac{3a}{4r}\left(1+\frac{a^2}{3r^2}\right) + \hat{\mathbf{e}}_r\,(\hat{\mathbf{e}}_r\cdot\mathbf{v}_S)\frac{3a}{4r}\left(1-\frac{a^2}{r^2}\right), \tag{A.13}$$

and the friction on the sphere

$$\mathbf{F} = -\zeta \mathbf{v}_S = -6\pi\mu a \mathbf{v}_S. \tag{A.14}$$

F is known as the Stokes friction.

A.3 Hydrodynamic interactions between slowly moving spheres

In the previous subsection we calculated the flow field in the solvent around a *single* slowly moving sphere. When more than one sphere is present in the system, this flow field will be felt by the other spheres. As a result these spheres experience a force which is said to result from hydrodynamic interactions with the original sphere.

We will assume that at each time the fluid flow field can be treated as a steady state flow field [15]. This is true for very slow flows, where changes in positions and velocities of the spheres take place over much larger time scales than the time it takes for the fluid flow field to react to such changes. The hydrodynamic problem then is to find a flow field satisfying the stationary Stokes equations,

$$\mu\nabla^2\mathbf{v} = \boldsymbol{\nabla} P \tag{A.15}$$

$$\boldsymbol{\nabla}\cdot\mathbf{v} = 0, \tag{A.16}$$

together with the boundary conditions

$$\mathbf{v}(\mathbf{R}_i + \mathbf{a}) = \mathbf{v}_i \qquad \forall i, \tag{A.17}$$

where $\mathbf{R}_i$ is the position vector and $\mathbf{v}_i$ is the velocity vector of the i'th sphere, and **a** is any vector of length a. If the spheres are very far apart we may approximately consider any one of them to be alone in the fluid. The flow field is then just the sum of all flow fields emanating from the different spheres

$$\mathbf{v}(\mathbf{r}) = \sum_i \mathbf{v}_i^{(0)}(\mathbf{r}-\mathbf{R}_i), \tag{A.18}$$

where, according to Eq. (A.13),

$$\begin{aligned}\mathbf{v}_i^{(0)}(\mathbf{r}-\mathbf{R}_i) = \mathbf{v}_i \frac{3a}{4|\mathbf{r}-\mathbf{R}_i|}\left[1+\frac{a^2}{3(\mathbf{r}-\mathbf{R}_i)^2}\right] \\ +(\mathbf{r}-\mathbf{R}_i)((\mathbf{r}-\mathbf{R}_i)\cdot\mathbf{v}_i)\frac{3a}{4|\mathbf{r}-\mathbf{R}_i|^3}\left[1-\frac{a^2}{(\mathbf{r}-\mathbf{R}_i)^2}\right].\end{aligned} \tag{A.19}$$

We shall now calculate the correction to this flow field, which is of lowest order in the sphere separation.

We shall first discuss the situation for only two spheres in the fluid. In the neighbourhood of sphere one the velocity field may be written as

$$\mathbf{v}(\mathbf{r}) = \mathbf{v}_1^{(0)}(\mathbf{r}-\mathbf{R}_1) + \frac{3a}{4|\mathbf{r}-\mathbf{R}_2|}\left[\mathbf{v}_2 + \frac{(\mathbf{r}-\mathbf{R}_2)}{|\mathbf{r}-\mathbf{R}_2|}\frac{(\mathbf{r}-\mathbf{R}_2)}{|\mathbf{r}-\mathbf{R}_2|}\cdot\mathbf{v}_2\right], \tag{A.20}$$

where we have approximated $\mathbf{v}_2^{(0)}(\mathbf{r}-\mathbf{R}_2)$ to terms of order $a/|\mathbf{r}-\mathbf{R}_2|$. On the surface of sphere one we approximate this further by

$$\mathbf{v}(\mathbf{R}_1+\mathbf{a}) = \mathbf{v}_1^{(0)}(\mathbf{a}) + \frac{3a}{4R_{21}}\left(\mathbf{v}_2 + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\cdot\mathbf{v}_2\right), \tag{A.21}$$

where $\hat{\mathbf{R}}_{21} = (\mathbf{R}_2-\mathbf{R}_1)/|\mathbf{R}_2-\mathbf{R}_1|$. Because $\mathbf{v}_1^{(0)}(\mathbf{a}) = \mathbf{v}_1$, we notice that this result is not consistent with the boundary condition $\mathbf{v}(\mathbf{R}_1+\mathbf{a}) = \mathbf{v}_1$. In order to satisfy this boundary condition we subtract from our results so far, a solution of Eqs. (A.15) and (A.16) which goes to zero at infinity, and which on the surface of sphere one corrects for the second term in Eq. (A.21). The flow field in the neighbourhood of sphere one then reads

$$\begin{aligned}
\mathbf{v}(\mathbf{r}) &= \mathbf{v}_1^{\mathrm{corr}}\frac{3a}{4|\mathbf{r}-\mathbf{R}_1|}\left[1+\frac{a^2}{3(\mathbf{r}-\mathbf{R}_1)^2}\right] \\
&\quad + (\mathbf{r}-\mathbf{R}_1)((\mathbf{r}-\mathbf{R}_1)\cdot\mathbf{v}_1^{\mathrm{corr}})\frac{3a}{4|\mathbf{r}-\mathbf{R}_1|^3}\left[1-\frac{a^2}{(\mathbf{r}-\mathbf{R}_1)^2}\right] \\
&\quad + \frac{3a}{4R_{21}}\left(\mathbf{v}_2 + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\cdot\mathbf{v}_2\right) \qquad \text{(A.22)} \\
\mathbf{v}_1^{\mathrm{corr}} &= \mathbf{v}_1 - \frac{3a}{4R_{21}}\left(\mathbf{v}_2 + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\cdot\mathbf{v}_2\right). \qquad \text{(A.23)}
\end{aligned}$$

The flow field in the neighbourhood of sphere two is treated similarly.

We notice that the correction that we have applied to the flow field in order to satisfy the boundary conditions at the surface of sphere one is of order a/R_{21}. Its strength in the neighbourhood of sphere two is then of order $(a/R_{21})^2$, and need therefore not be taken into account when the flow field is adapted to the boundary conditions at sphere two.

The flow field around sphere one is now given by Eqs. (A.22) and (A.23). The last term in Eq. (A.22) does not contribute to the stress tensor (the gradient of a constant field is zero). The force exerted by the fluid on sphere one then equals $-6\pi\mu a\mathbf{v}_1^{\mathrm{corr}}$. A similar result holds for sphere two. In full we have

$$\begin{aligned}
\mathbf{F}_1 &= -6\pi\mu a\mathbf{v}_1 + 6\pi\mu a\frac{3a}{4R_{21}}\left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\right)\cdot\mathbf{v}_2 \qquad \text{(A.24)} \\
\mathbf{F}_2 &= -6\pi\mu a\mathbf{v}_2 + 6\pi\mu a\frac{3a}{4R_{21}}\left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\right)\cdot\mathbf{v}_1, \qquad \text{(A.25)}
\end{aligned}$$

where $\bar{\mathbf{I}}$ is the three-dimensional unit tensor. Inverting these equations, retaining only terms up to order a/R_{21}, we get

$$\mathbf{v}_1 = -\frac{1}{6\pi\mu a}\mathbf{F}_1 - \frac{1}{8\pi\mu R_{21}}\left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\right)\cdot\mathbf{F}_2 \tag{A.26}$$

$$\mathbf{v}_2 = -\frac{1}{6\pi\mu a}\mathbf{F}_2 - \frac{1}{8\pi\mu R_{21}}\left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{21}\hat{\mathbf{R}}_{21}\right)\cdot\mathbf{F}_1 \tag{A.27}$$

When more than two spheres are present in the fluid, corrections resulting from n-body interactions ($n \geq 3$) are of order $(a/R_{ij})^2$ or higher and need not be taken into account. The above treatment therefore generalizes to

$$\mathbf{F}_i = -\sum_{j=0}^{N} \bar{\boldsymbol{\zeta}}_{ij}\cdot\mathbf{v}_j \tag{A.28}$$

$$\mathbf{v}_i = -\sum_{j=0}^{N} \bar{\boldsymbol{\mu}}_{ij}\cdot\mathbf{F}_j, \tag{A.29}$$

where

$$\bar{\boldsymbol{\zeta}}_{ii} = 6\pi\mu a\bar{\mathbf{I}}, \qquad \bar{\boldsymbol{\zeta}}_{ij} = -6\pi\mu a\frac{3a}{4R_{ij}}\left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{ij}\hat{\mathbf{R}}_{ij}\right) \tag{A.30}$$

$$\bar{\boldsymbol{\mu}}_{ii} = \frac{1}{6\pi\mu a}\bar{\mathbf{I}}, \qquad \bar{\boldsymbol{\mu}}_{ij} = \frac{1}{8\pi\mu R_{ij}}\left(\bar{\mathbf{I}} + \hat{\mathbf{R}}_{ij}\hat{\mathbf{R}}_{ij}\right). \tag{A.31}$$

$\bar{\boldsymbol{\mu}}_{ij}$ is generally called the mobility tensor. The specific form Eq. (A.31) is known as the Oseen tensor.

APPENDIX B

MATHEMATICAL RELATIONS

B.1 Gaussian integrals

We will often need to evaluate integrals of the form

$$I_n(\alpha) = \int_0^\infty \mathrm{d}x\, x^n \mathrm{e}^{-\alpha x^2}. \tag{B.1}$$

Solutions to these integrals can easily be generated by "differentiating under the integral sign":

$$I_{n+2}(\alpha) = \int_0^\infty \mathrm{d}x\, x^{n+2} \mathrm{e}^{-\alpha x^2} = -\frac{\partial}{\partial \alpha} I_n(\alpha). \tag{B.2}$$

Knowledge of the solution for $n = 0$ allows us to generate solutions for all even n, whereas the solution for $n = 1$ allows us to generate solutions for all odd n.

Although the solution for $n = 1$ can easily be found, $I_1(\alpha) = 1/(2\alpha)$, the solution for $n = 0$ requires some more thought. It can be calculated by considering the following integral in the two-dimensional plane:

$$\int_0^\infty \mathrm{d}x \int_0^\infty \mathrm{d}y\, \mathrm{e}^{-\alpha(x^2+y^2)}. \tag{B.3}$$

Because we can factorize the exponential this is clearly equal to $[I_0(\alpha)]^2$. Changing from (x, y) to polar coordinates (r, ϕ), taking into account the Jacobian, we find

$$I_0^2(\alpha) = \int_0^\infty \mathrm{d}r \int_0^{\pi/2} \mathrm{d}\phi\, r \mathrm{e}^{-\alpha r^2} = \frac{\pi}{4\alpha}. \tag{B.4}$$

Hence $I_0(\alpha) = \frac{1}{2}\sqrt{\frac{\pi}{\alpha}}$.

The integrals I_n appear so often that we write down the values of some of them explicitly. We have

$$I_0 = \frac{1}{2}\left(\frac{\pi}{\alpha}\right)^{1/2}, \quad I_2 = \frac{1}{4}\left(\frac{\pi}{\alpha^3}\right)^{1/2}, \quad I_4 = \frac{3}{8}\left(\frac{\pi}{\alpha^5}\right)^{1/2}, \ldots \tag{B.5}$$

and

$$I_1 = \frac{1}{2\alpha}, \quad I_3 = \frac{1}{2\alpha^2}, \quad I_5 = \frac{1}{\alpha^3}, \ldots. \tag{B.6}$$

B.2 Geometric series

The geometric series $S_n(x)$ is defined as

$$S_n(x) = \sum_{k=0}^{n} x^n = 1 + x + x^2 + \ldots + x^n. \tag{B.7}$$

Multiplying both sides by x we find

$$xS_n(x) = x + x^2 + x^3 + \ldots + x^{n+1}. \tag{B.8}$$

Subtracting these two equations yields

$$(1-x)S_n(x) = 1 - x^{n+1}. \tag{B.9}$$

So we find

$$\sum_{k=0}^{n} x^n = \frac{1-x^{n+1}}{1-x}. \tag{B.10}$$

For $-1 < x < 1$ the series converges as $n \to \infty$, in which case

$$\sum_{k=0}^{\infty} x^n = \frac{1}{1-x} \qquad (|x| < 1). \tag{B.11}$$

B.3 Taylor series

A Taylor series is a series expansion of a function around a certain point. In one dimension, the Taylor series of a function f around the point $x = a$ is given by

$$f(x) = f(a) + f'(a)(x-a) + \frac{1}{2}f''(a)(x-a)^2 + \frac{1}{6}f'''(a)(x-a)^3 + \ldots + \frac{1}{n!}f^{(n)}(a)(x-a)^n + \ldots, \tag{B.12}$$

where $f^{(n)}(a)$ denotes the n-th derivative of f at $x = a$.

Here follow a few examples:

$$\exp(x) = 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3 + \ldots + \frac{1}{n!}x^n + \ldots \tag{B.13}$$

$$\ln(1+x) = x - \frac{1}{2}x^2 + \frac{1}{3}x^3 + \ldots + \frac{(-1)^{n-1}}{n}x^n + \ldots \tag{B.14}$$

$$\cos(x) = 1 - \frac{1}{2}x^2 + \frac{1}{24}x^4 + \ldots + \frac{\cos(n\pi/2)}{n!}x^n + \ldots \tag{B.15}$$

$$\sin(x) = x - \frac{1}{6}x^3 + \frac{1}{120}x^5 + \ldots + \frac{\sin(n\pi/2)}{n!}x^n + \ldots \tag{B.16}$$

B.4 Logarithms and exponentials

When manipulating logarithms and exponentials remember the following rules:

$$\exp(a+b) = \exp(a)\exp(b) \tag{B.17}$$

$$\exp(a\ln b) = b^a \tag{B.18}$$

$$\exp(ab) = (\exp(a))^b = (\exp(b))^a \tag{B.19}$$

$$\ln(ab) = \ln(a) + \ln(b) \tag{B.20}$$

$$\ln(b^a) = a\ln(b) \tag{B.21}$$

Derivatives are given by

$$\frac{\mathrm{d}}{\mathrm{d}x} b\exp(ax) = ab\exp(ax) \tag{B.22}$$

$$\frac{\mathrm{d}}{\mathrm{d}x} a^x = \ln(a)\, a^x \tag{B.23}$$

$$\frac{\mathrm{d}}{\mathrm{d}x} b\ln(cx^a) = b\frac{a}{x} \tag{B.24}$$

and integrals by

$$\int \exp(ax)\mathrm{d}x = \frac{1}{a}\exp(ax) \tag{B.25}$$

$$\int \ln(ax)\mathrm{d}x = x\ln(ax) - x \tag{B.26}$$

APPENDIX C

RANDOM NUMBER GENERATORS

C.1 Uniform random numbers

Most modern programming languages include a function that generates numbers ξ that are uniformly distributed in the range $\xi \in (0,1)$. The generated numbers are not truly random, but *pseudo-random*: successive numbers are generated by aritmetic manipulation inside the computer. The trick is to produce a repeatable sequence that passes a wide range of statistical tests for independence [4]. We will assume that you have available such a uniform random number generator. Usually they are called something like `RAN`. Be careful: functions that generate a random integer between 0 and a maximum integer often have similar names. For example, in C the function `rand()` returns a random integer between 0 and `RAND_MAX`. This may be used to make a quick and dirty uniform random number generator: $\xi =$ `rand()/ double(RAND_MAX)` .

C.2 Gaussian random numbers

For several applications we need numbers from a Gaussian (or normal) distribution with zero average:

$$P(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{x^2}{2\sigma^2}\right], \tag{C.1}$$

where σ^2 is the variance and σ the standard deviation of the distribution.

Here we describe two methods to generate a Gaussian random number with unit variance. Note that a Gaussian number with variance σ^2 is generated by simply multiplying the resulting random number x by σ. The first method is called the Box-Muller transform [13]:

- generate two uniform random numbers: ξ_1 and ξ_2;
- calculate $x_1 = (-2\ln\xi_1)^{1/2}\cos(2\pi\xi_2)$ and $x_2 = (-2\ln\xi_1)^{1/2}\sin(2\pi\xi_2)$.

The numbers x_1 and x_2 are then two independent Gaussian random numbers. The second method involves the generation of 12 uniform random numbers:

- generate 12 uniform random numbers $\xi_1, \ldots, \xi_{12}$;
- calculate $x = \sum_{i=1}^{12} \xi_i - 6$.

This generates an approximately Gaussian random number (by virtue of the central limit theorem). Clearly random numbers outside the range $(-6, 6)$ will never occur, but for some applications (such as initialising velocities in a simulation) this may actually be an advantage. The computational speed relative to the Box-Muller technique will depend on the timing of the logarithm, square root, cosine and sine functions, as well as the uniform random number generator itself.

C.3 Constructing random numbers with other distributions

Given an analytical expression $P(x)$ for some normalised distribution between x_{min} and x_{max} (possibly $\pm\infty$), it is in some cases possible to construct an analytical mapping between a uniform random number ξ and a number x from $P(x)$. First determine the cummulative probability up to x:

$$C(x) \equiv \int_{x_{min}}^{x} P(x')\mathrm{d}x'. \tag{C.2}$$

Because $P(x)$ is normalised, $C(x)$ ranges from 0 at x_{min} to 1 at x_{max}. Indeed, $C(x)$ presents the mapping between x and the uniform distribution. If $C(x)$ can be inverted, we have the mapping from a uniform distribution to the distribution $P(x)$:

$$x = C^{-1}(\xi). \tag{C.3}$$

As an explicit example, consider the biased velocity distribution in Eq. (2.21),

$$P(v) = \frac{m}{k_B T} v \mathrm{e}^{-\frac{mv^2}{2k_B T}}. \tag{C.4}$$

The cummulative probability is:

$$C(v) = \int_0^v P(v')\mathrm{d}v' = 1 - \mathrm{e}^{-\frac{mv^2}{2k_B T}} \tag{C.5}$$

Equating $C(v) = \xi$ and inverting, we find our mapping:

$$v = \left(-\frac{2k_B T}{m}\ln(1-\xi)\right)^{1/2}. \tag{C.6}$$

APPENDIX

PHYSICAL CONSTANTS

Planck's constant, $h = 6.62607 \times 10^{-34}$ J s
atomic mass unit, $u = 1.66054 \times 10^{-27}$ kg
Boltzmann's constant, $k_B = 1.38065 \times 10^{-23}$ J/K
Avogadro's number, $N_{Av} = 6.02214 \times 10^{23}$ mol^{-1}
Universal gas constant, $R = 8.3145$ J/K mol^{-1}
Permittivity of vacuum, $\epsilon_0 = 8.854 \times 10^{-12}$ F/m

BIBLIOGRAPHY

[1] R. Adhikari, K. Stratford, M.E. Cates, and A.J. Wagner, *Fluctuating lattice Boltzmann*, Europhys. Lett. **71**, 473 (2005).

[2] B.J. Alder and T.E. Wainwright, *Studies in Molecular Dynamics. I. General Method*, Journal of Chemical Physics **31**, 459 (1959).

[3] E. Allahyarov and G. Gompper, Phys. Rev. E **66**, 036702 (2002).

[4] M.P. Allen and D.J. Tildesley, *Computer Simulation of Liquids*, Clarendon (Oxford), 1987.

[5] S. Asakura and F. Oosawa, J. Polym. Sci., Polym. Symp. **33**, 183 (1958); A. Vrij, Pure Appl. Chem. **48**, 471 (1976).

[6] J.-L. Barrat, L. Bocquet, Phys. Rev. Lett. **82**, 4671 (1999).

[7] G. K. Batchelor, J. Fluid Mech. **56**, 375 (1972).

[8] R. Beetstra, M.A. van der Hoef and J.A.M. Kuipers, *Drag force from lattice Boltzmann simulations of intermediate Reynolds number flow past mono- and bidisperse arrays of spheres*, A.I.Ch.E. Journal **53**, 489 (2007).

[9] H.C. Berg, *Random Walks in Biology*, Princeton University Press (Princeton, 1993).

[10] G.A. Bird, *Molecular Gas Dynamics and the Direct Simulation of Gas Flows*, Clarendon (Oxford), 1994.

[11] L. Bocquet and J.-L. Barrat, Phys. Rev. E **49**, 3079 (1994).

[12] G.A. Bokkers, *Multi-level modeling of the hydrodynamics in gas phase polymerisation reactors*, Ph.D. thesis, University of Twente, The Netherlands, 2005.

[13] G.E.P. Box and M.E. Muller, *A note on the generation of random normal deviates*, Ann. Math. Stat. **29**, 610 (1958).

[14] J.F. Brady and G. Bossis, *Stokesian Dynamics*, Ann. Rev. Fluid Mech. **20**, 111 (1988).

[15] W.J. Briels, *Theory of Polymer Dynamics*, Lecture notes, Uppsala (1994).

[16] P.A. Cundall and O.D.L. Strack, *A discrete numerical model for granular assemblies*, Geotechnique **29**, 47 (1979).

[17] N.G. Deen, *Review of discrete particle modeling of fluidized beds*, Chem. Eng. Sci. **62**, 28 (2007).

[18] E. Delnoij, J.A.M. Kuipers and W.P.M. van Swaaij, *A three-dimensional CFD model for gas-liquid bubble columns*, Chem. Eng. Sci. **54**, 2217 (1999).

[19] J. K. G. Dhont, *An Introduction to the Dynamics of Colloids*, Elsevier, (Amsterdam, 1996).

[20] B. Dünweg, U.D. Schiller, and A.J.C. Ladd, *Statistical mechanics of the fluctuating lattice Boltzmann equation*, Phys. Rev. E **76**, 036704 (2007).

[21] J. Dzubiella, H. Löwen, and C.N. Likos, Phys. Rev. Lett. **91**, 248301 (2003).

[22] S. Ergun, *Fluid flow through packed columns*, Chem. Eng. Progress **48**, 89 (1952).

[23] P. Español and P.B. Warren, *Statistical Mechanics of Dissipative Particle Dynamics*, Europhys. Lett. **30**, 191 (1995).

[24] G. Gompper, T. Ihle, D.M. Kroll and R.G. Winkler, *Multi-Particle Collision Dynamics: A Particle-Based Mesoscale Simulation Approach to the Hydrodynamics of Complex Fluids*, Adv. Polymer Sci. **221**, 1 (2009).

[25] R.D. Groot and P.B. Warren, *Dissipative Particle Dynamics: Bridging the gap between atomisticc and mesoscopic simulation*, J. Chem. Phys. **107**, 4423 (1997).

[26] E. Guyon, J.-P. Hulin, L. Petit and C. D. Mitescu, *Physical Hydrodynamics*, (Oxford Univeristy Press, Oxford, 2001).

[27] J.P. Hansen and I.R. McDonald, *Theory of Simple Liquids, 2nd Ed.*, Academic Press, London (1986).

[28] J. Happel and H. Brenner and *Low Reynolds Number Hydrodynamics : with special applications to particulate media*, Springer (New York 1983).

[29] M. Hecht, J. Harting, T. Ihle and H.J. Herrmann, *Simulation of claylike colloids*, Phys. Rev. E **72**, 011408 (2005).

[30] M.A. van der Hoef, M. Ye, M. van Sint Annaland, A.T. Andrews IV, S. Sundaresan and J.A.M. Kuipers, *Multiscale modeling of gas-fluidized beds*, Adv. Chem. Eng. **31**, 65 (2006).

[31] P.J. Hoogerburgge and J.M. Koelman, *Simulating microscopic hydrodynamic phenomena with dissipative particle dynamics*, Europhys. Lett. **19**, 155 (1992).

[32] B.P.B. Hoomans, *Granular dynamics of gas-solid two-phase flows*, PhD thesis, University of Twente, The Netherlands, 1999.

[33] T. Ihle and D. M. Kroll, Phys. Rev. E **63** 020201(R) (2001)

[34] T. Ihle and D. M. Kroll, Phys. Rev. E **67**, 066705 (2003); *ibid* 066706 (2003).

[35] T. Ihle, E. Tuzel, and D.M. Kroll, Phys. Rev. E, **72**, 046707 (2005).

[36] N. Kikuchi, C. M. Pooley, J. F. Ryder, and J. M. Yeomans, J. Chem. Phys. **119**, 6388 (2003).

[37] A. J. C. Ladd, Phys. Rev. Lett. **76**, 1392 (1996); *ibid*, **88**, 048301 (2002).

[38] A. Lamura, G. Gompper, T. Ihle, and D.M. Kroll, Europhys. Lett. **56**, 319 (2001).

[39] J.A. Laverman, *On the hydrodynamics in gas polymerization reactors*, PhD thesis, Eindhoven University of Technology, The Netherlands, 2010.

[40] C.P. Lowe, Europhys. Lett. **47**, 145 (1999).

[41] C.P. Lowe, A.F. Bakker and M.W. Dreischor, Europhys. Lett. **67**, 397 (2004).

[42] A. Malevanets and R. Kapral, *Mesoscopic model for solvent dynamics*, J. Chem. Phys. **110**, 8605 (1999).

[43] A. Malevanets and R. Kapral, J. Chem. Phys., **112**, 7260 (2000).

[44] J. C. Maxwell, in *The Scientific Papers of James Clerk Maxwell*, Vol. 2., W. D. Niven ed. (Dover, New York, 1965).

[45] D.A. McQuarrie, *Statistical Mechanics*, Harper & Row (New York, USA), 1976.

[46] J.R. Melrose and R.C. Ball, J. Rheol. **48**, 937 (2004).

[47] D.R. Mikulencak and J.F. Morris, J. Fluid. Mech. **520**, 215 (2004).

[48] A. Moncho-Jordá, A.A. Louis and J.T. Padding, *The effects of inter-particle attractions on colloidal sedimentation*, Phys. Rev. Lett. **104**, 068301 (2010).

[49] N.-Q. Nguyen and A. J. C. Ladd, Phys. Rev. Lett. **66**, 046708 (2002).

[50] J.T. Padding and A.A. Louis, *Hydrodynamic and Brownian Fluctuations in Sedimenting Suspensions*, Phys. Rev. Lett. **93**, 220601 (2004).

[51] J.T. Padding and A.A. Louis, *Hydrodynamic interactions and Brownian forces in colloidal suspensions: Coarse-graining over time and length scales*, Phys. Rev. E **74**, 031402 (2006).

[52] J.T. Padding and W.J. Briels, *Translational and rotational friction on a colloidal rod near a wall*, J. Chem. Phys. **132**, 054511 (2010).

[53] C.M. Pooley and J.M. Yeomans, J. Phys. Chem. B **109**, 6505 (2005).

[54] E. Purcell *Life at Low Reynolds Numbers*, Am. J. of Physics **45**, 3 (1977). See also http://brodylab.eng.uci.edu/j̃pbrody/reynolds/lowpurcell.html

[55] M. Ripoll, K. Mussawisade, R.G. Winkler, and G. Gompper, Phys. Rev. E **72**, 016701 (2005).

[56] W. B. Russell, D. A. Saville and W. R. Showalter, *Colloidal Dispersions*, Cambridge University Press, Cambridge (1989).

[57] A. Sierou and J.F. Brady, *Accelerated Stokesian Dynamics simulations*, J. Fluid Mech. **448**, 115 (2001).

[58] S. Succi, *The Lattice Boltzmann Equation for Fluid Dynamics and Beyond*, Oxford University Press, 2001.

[59] J. Vermant and M.J. Solomon, J. Phys.: Condens. Matt. **17**, R187 (2005).

[60] G. A. Vliegenthart and P. van der Schoot, Europhys. Lett. **62**, 600 (2003).

[61] Y.C. Wen and Y.H. Yu, *Mechanics of fluidization*, Chem. Eng. Progress Symposium Series **62**, 100 (1966).

INDEX