TU Delft

Fluid-Particle Simulations

CFD-DEM Coupling

Literature Study Report

Davide Fantin

March 18, 2018

Contents

Chapter 1

Introduction

Mathematical models and their simulations are becoming more and more fundamental in both industrial and academical environments. Thanks to the increasing computational capabilities, it is nowadays possible to simulate industrial and physical processes, exploring their evolution in time without actually performing them.

The field of numerical simulations has an extremely wide set of possible applications. Engineering problems can be faced with the help of computer simulations, which can provide fast and reliable results in complex situations. Design products optimizing specific physical properties, exploring scenarios, predicting vulnerabilities are just a few of the cases in which a numerical approach can be as efficient as low-cost.
Applications of simulations have been developed in a huge variety of fields, both in fundamental academic research and in industrial applications, e.g. petrochemical, marine, pharmaceutical and aeronautical industries.

The main feature that numerical simulations provide is optimal and low-cost development of products and services. To this purpose, simulations of fluid/fluid, fluid/solid particles systems are essential tools and require deep research to obtain realistic results.

Using a "mathematical" perspective, it is possible to claim that *at a first order approximation* the entire world is based on fluids/particles interactions.
This is the reason why one of the most challenging fields is the study of fluid/particles sytems, which has applications in uncountable different contextes. The fluid behavior is described by fluid dynamics, more precisely by Navier-Stokes equations and in this work it is simulated through methods of Computational Fluid Dynamics (CFD), whereas the dynamics of discrete solids or particles is described using Newton's law on each particle and, in this work, simulated using Discrete Element Method (DEM).

The aim of this work is to develop a robust coupling between an open source CFD toolbox (OpenFOAM) and DEM.

Chapter 2

Problem Description

Obtaining accurate and reliable results from simulations of fluid-particle systems is extremely important, therefore in the last decades academical and industrial researches focused more and more on the attempt to develop efficient numerical approaches to be used in all kinds of applications.

Numerical methods and their implementations are continuously improved, year by year, and the literature on mathematical modeling and numerical simulations has become extensive. Even if progress has grown exponentially over the years, a lot of work is still to be done, both in the modeling and in the simulation.

As stated in the introduction, the aim of this work is to study the coupling between fluid simulations, through the methods of CFD, implemented in OpenFOAM, and solid particle interactions, modeled by DEM.
The current implementation in OpenFOAM already includes a rudimentary interaction between the modeled particles themselves and between the particles and flow, but it has significant limitations on the size and shape of the modeled particles. In fact, particles are only approximated as spheres, but the most problematic aspect of the implementation is the ratio between particles and the fluid grid.
Therefore, the objective is to develop a new and robust coupling algorithm to improve the current implementation.

To this purpose, a literature study has been carried out, to develop the necessary basis to understand the state-of-the-art approaches adopted to simulate fluid/particle systems. This report is the result of the work and it provides the most relevant ideas that have been developed in the field.

Firstly, in Chapter 3 the equations of incompressible fluid dynamics are presented and derived. A brief introduction to the open source software OpenFOAM is given. Then, in Chapter 4 the Discrete Element Method for modeling particle/particle interactions is described and two softwares packages for DEM simulation are presented: HADES and LIGGGHTS, respectively. In Chapter 5 the state-of-the-art techniques for the coupling between CFD and DEM are described in detail. Finally, in Chapter 6 possible directions for the MSc project are briefly listed.

.

Chapter 3

Incompressible Fluid Dynamics

In this chapter fluid flows are considered. We briefly introduce the equations that arise from incompressible fluid dynamics. The Navier-Stokes equations are derived under the hypothesis of incompressibility. Then, OpenFOAM, an open source software package for fluid simulations, is introduced.

3.1 Derivation of Incompressible Navier-Stokes equations

Derivation of Navier Stokes equations appears in every monograph dedicated to fluid and flows, therefore in this report only main ideas are presented. We follow the derivation of the equation in [11]

The first step in the derivation of Navier Stokes equations is to apply the principle of mass conservation to a control volume $V(t)$ that contains a specific collection of fluid particles.
Let $\rho(\mathbf{x}, t)$ be the density of the fluid at point $\mathbf{x}$ at time t, then conservation of mass applied on control volume $V(t)$ reads:

$$\frac{d}{dt}\int_{V(t)} \rho(\mathbf{x}, t) dV = 0. \tag{3.1}$$

Let $\mathbf{u}(\mathbf{x}, t)$ be the velocity field, $A(t)$ the surface of the control volume and n the outward normal to the surface. Then applying Reynolds transport theorem we get:

$$\int_{V(t)} \frac{\partial}{\partial t}\rho(\mathbf{x}, t) dV + \int_{A(t)} \rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) \cdot \mathbf{n} dA = 0. \tag{3.2}$$

Applying divergence theorem to the surface integral we get:

$$\int_{V(t)} \left\{ \frac{\partial}{\partial t}\rho(\mathbf{x}, t) + \nabla \cdot (\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)) \right\} dV = 0. \tag{3.3}$$

Equation (3.3) has to be valid for all possible control volumes, in particular for vanishing control volumes. Therefore, the integrand in the left hand side has to be zero. We derived the differential form of the mass conservation, or the *continuity equation*.

$$\frac{\partial}{\partial t}\rho(\mathbf{x}, t) + \nabla \cdot (\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)) = 0. \tag{3.4}$$

Usually, the continuity equation (3.4) is written as:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \tag{3.5}$$

Adding the hypothesis of incompressible flows, equation (3.5) becomes:

$$\nabla \cdot \mathbf{u} = 0. \tag{3.6}$$

We will follow the same procedure applying the principle of momentum conservation to the control volume $V(t)$. The conservation of momentum deals with forces applied to the control volume. We will divide the forces in two contributes: *body forces*, that act without physical contact with the element and *surface forces*, that act through direct contact with the surface of the element. To this purpose, let $\mathbf{f}(\mathbf{x}, t)$ be the body force per unit mass on the fluid inside $V(t)$ and let $\mathbf{t}(\mathbf{n}, \mathbf{x}, t)$ be the surface force per unit area on the surface $A(t)$, usually called *stress vector*.

$$\frac{d}{dt}\int_{V(t)} \rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)dV = \int_{V(t)} \rho(\mathbf{x}, t)\mathbf{f}(\mathbf{x}, t)dV + \int_{A(t)} \mathbf{t}(\mathbf{n}, \mathbf{x}, t)dA \tag{3.7}$$

We use the assumption on curvature of Cauchy, i.e. $\mathbf{t}(\mathbf{n}, \mathbf{x}, t) = \mathbf{T}(\mathbf{x}, t)\mathbf{n}$, where $\mathbf{T}(\mathbf{x}, t)$ is usually called *stress tensor*. This corresponds to the hypothesis that the stress vector is a linear function of the stress tensor and the normal derivative to the surface. Applying the Reynolds Transport Theorem and the divergence theorem on the surface integral we get:

$$\int_{V(t)} \left\{ \frac{\partial}{\partial t}\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) + \nabla \cdot (\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)) - \rho(\mathbf{x}, t)\mathbf{f}(\mathbf{x}, t) - \nabla \cdot (\mathbf{T}(\mathbf{x}, t)) \right\} dV = 0. \tag{3.8}$$

As the case of mass conservation, equation (3.8) has to hold for vanishing control volumes, therefore the integrand on the left hand size has to be zero.

$$\frac{\partial}{\partial t}\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t) + \nabla \cdot (\rho(\mathbf{x}, t)\mathbf{u}(\mathbf{x}, t)) - \rho(\mathbf{x}, t)\mathbf{f}(\mathbf{x}, t) - \nabla \cdot (\mathbf{T}(\mathbf{x}, t)) = 0. \tag{3.9}$$

Using the continuity equation (3.4) and the shorter notation we can simplify the equation (3.9) and we obtain the *Cauchy equation of motion*:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = \nabla \cdot \mathbf{T} + \mathbf{f}. \tag{3.10}$$

Equation (3.10) is able to describe the conservation of momentum at differential level for both fluids and solids. In order to describe fluids we have to consider a constitutive equation for the stress tensor $\mathbf{T}$.
We consider *newtonian* fluids, in which the stress tensor $\mathbf{T}$ is a linear function of the rate of strain tensor $\mathbf{E} = \frac{1}{2}(\nabla \mathbf{u} + \nabla \mathbf{u})^T$ and we add the *incompressibility* assumption. Hence, we get the following equation for the stress tensor $\mathbf{T}$:

$$\mathbf{T} = -p\mathbf{I} + 2\mu\mathbf{E}, \tag{3.11}$$

where p is the pressure and μ is the viscosity of the fluid.
Substituing the constituve relation on the Cauchy equation of motion (3.10) gives:

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\nabla p + \mu\Delta \mathbf{u} + \mathbf{f}. \tag{3.12}$$

Considering the continuity equation (3.6) and the Cauchy equation for incompressible newtonian fluids (3.12) we finally get the *Navier Stokes equations*:

$$\begin{cases} \nabla \cdot \mathbf{u} = 0 \\ \rho\left(\dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u}\right) = -\nabla p + \mu\Delta \mathbf{u} + \mathbf{f}. \end{cases} \tag{3.13}$$

3.2 OpenFOAM

The Navier Stokes equations derived in the previous section are a system of non-linear partial differential equations. They manage to describe the behavior of general 3D flows, laminar or turbulent. They are extremely efficient in modeling all kind of flows but, due to non-linearities, an analytic solution is not available for the general case.
Therefore, up to now, only numerical solutions are available. Numerical solutions are determined by discretization of the equations and on development of algorithms to get an approximation of the solution of the original (continous) problem.

Several commercial software packages are able to simulate efficiently the behavior of fluids in several different situations. In this report we will consider an open source software, OpenFOAM and we will refer to its user's guide [6]. OpenFOAM (*"Open source Field Operation And Manipulation"*) is a C++ toolbox for the development of customized numerical solvers for the solution of continuum mechanics problems. It also provides pre- and post-processing utilities.
Basically, OpenFOAM is a library, which can be used to build the so called *applications*. Applications can be *solvers* or *utilities*. Solvers perform the calculation to solve a specific problem. Utilities prepare the mesh, set-up the simulation case and process the results.

OpenFOAM uses Finite Volume Method (FVM) for the discretization and the solution of partial differential equations. The idea is that the domain is divided in control volumes. On each control volume, partial differential equations are discretized and solved.
A dissertation of the FVM is beyond the scope of this work. Details of the implementation of the method and of the various techniques developed over the years can be found in monographs like [18] or [5].

OpenFOAM solvers have been developed for a broad set of problems. Some areas in which standard solvers available for fluid mechanics are: basic CFD, incompressible/compressible flows with DNS, RANS and LES capabilities, multiphase flows and particle-tracking solvers.
Other fields in which OpenFOAM has been used are: combustion, conjugate heat transfer, molecular dynamics, electromagnetism and solid dynamics.

Main built-in solvers
Several solvers have been developed for different applications. Here we list the most relevant to the purpose of this literature study.

Incompressible Flows:

- `icoFoam` Transient solver for incompressible, laminar flow of Newtonian fluids.
- `pimpleFoam` Large time-step transient solver for incompressible, turbulent flow, using the PIMPLE (merged PISO-SIMPLE) algorithm.
- `pimpleDyMFoam` Transient solver for incompressible, turbulent flow of Newtonian fluids on a moving mesh, with possibility of local refinements.
- `pisoFoam` Transient solver for incompressible, turbulent flow, using the PISO algorithm.
- `simpleFoam` Steady-state solver for incompressible, turbulent flow, using the SIMPLE algorithm.

Multiphase flows:

- `interFoam` Solver for 2 incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach.
- `interDyMFoam` Solver for 2 incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach, with optional mesh motion and mesh topology changes including adaptive re-meshing.
- `multiphaseEulerFoam` Solver for a system of many compressible fluid phases including heat-transfer.
- `multiphaseInterFoam` Solver for n incompressible fluids which captures the interfaces and includes surface-tension and contact-angle effects for each phase.

Direct numerical simulations

- `dnsFoam` Direct numerical simulation solver for boxes of isotropic turbulence.

Particle-tracking flows

- `DPMFoam` Transient solver for the coupled transport of a single kinematic particle cloud including the effect of the volume fraction of particles on the continuous phase.
  A more complete discussion of this solver is provided in section 5.6
- `MPPICFoam` Transient solver for the coupled transport of a single kinematic particle cloud including the effect of the volume fraction of particles on the continuous phase.
  Multi-Phase Particle In Cell (MPPIC) modeling is used to represent collisions without resolving particle-particle interactions. They are in fact represented by models which evaluate mean values calculated on the Eulerian mesh. A severe limitation of the solver is that the size of particles must be small compared to the Eulerian grid for accurate interpolation. This coupling will be defined as unresolved coupling in the following chapter.
  The solver provides reliable results in dense particle flows (more than 5% by volume), but since it does not resolve particle-particle interactions it is not useful for the aim of this work.

How to use the built-in solvers

Each simulation takes place in a specific directory, created by the user. In this directory there have to be three subdirectories:

- `system`
  It contains files to control the generation of the mesh and the integration method used to solve the specific problem.
- `constant`
  It contains the specification of constant of the problem and it store the mesh, once it has been generated following the instructions contained in the system directory
- `0`
  It contains one file per variable of the problem. Each file contains initial and boundary conditions for the specific variable.

All the files required by the simulation are text files with an appropriate syntax. Usually, two commands are necessary to perform a simulation: `blockMesh` generates the mesh and then a solver is used to actually perform the calculation, i.e. if we want to use icoFoam solver, it is

enough to use the command `icoFoam`. Other commands may be required in specific cases.

Once the simulation has been performed, other directories will be created in the directory of the project: OpenFOAM will store the evolution of the solution in time with an user-defined time step, usually larger than the actual time step used in the calculations. Each directory will contain one file per variable, and in that file the numerical values for that variable are stored. Usually the visualization of the results is done via ParaView, an open source multiple-platform application for interactive, scientific visualization.

A very first example of the output that can be obtained using OpenFOAM is given in Figure 3.1. We consider an isothermal, incompressible, laminar flow in a 2D square domain. Left, right and down side of the square are walls, at the top the fluid is moved with velocity 1 in the right direction. The simulation has been performed using a tutorial provided with OpenFOAM. Note that OpenFOAM solves three dimensional problems, even if the problem considered is two dimensional.

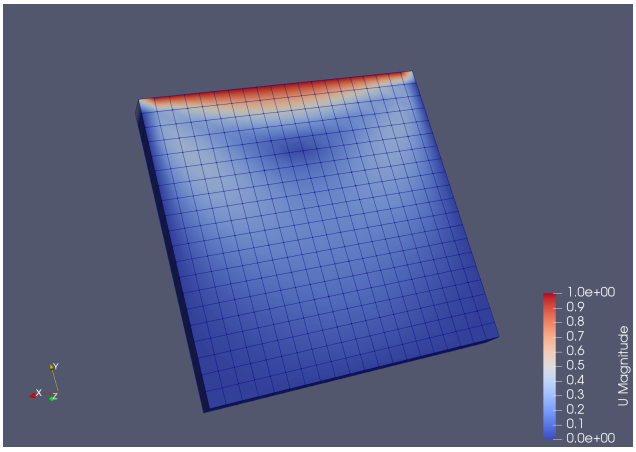

Figure 3.1: **Isothermal, Incompressible, Laminar flow in a 2D square domain** Left, right and down side of the square are walls, at the top the fluid is moved with velocity 1 in the right direction. Magnitude of velocity is plotted.

Chapter 4

Particle/Particle interactions

In this chapter, the interaction between particles in pure granular flows is considered.

Two different approaches are possible to describe particle/particle interaction: Lagrangian tracking or Eulerian modeling approaches.

- Lagrangian approach: individual particles (or parcels of) are tracked through the field and properties of each particle are evaluated.
  examples: Discrete Element Method (DEM), Discrete Parcle Method (DPM)

- Eulerian approach: sets of algebraic conservation equations are solved simultaneously for each node in the field.
  examples: Two Flow Model (TFM)

In this work, we focus on the description of Discrete Element Method. Firstly, we introduce the method. Then, two open source software packages, HADES and LIGGGHTS, that apply DEM are introduced and presented.

General overview of DEM
DEM is based on the assumption that the material in consideration is made of separate, discrete particles. Granular matter, bulk material, solutions, liquids, powder, rock masses are the most common example of application of DEM.
The first steps of development of a DEM for a particles system are the generation of a model, the orientation in space of all the particles and assignment of an initial velocity to them. The forces applied on each particle are computed from the initial data and depend on the model used to describe contact between particles and on the physical laws relevant on the specific problem. Possible important contributions can be given by:

- Macroscopic: Friction, Contact plasticity, Attractive potentials (cohesions, adhesion), Gravity

- Microscopic: Electrostatic attraction (Coulomb), intra-molecular forces (Van der Waals)

Every force taken into consideration for a specific problem is summed to have the total force exerted on every particle.

Once the total force acting on each particle is computed, it is possible to perform an integration in time to evaluate the new positions and the velocities of the particles, using suitable integration method, as the Verlet algorithm, symplectic integrators or the leapfrog method. The

simulation consists in applying this steps until a suitable final time is reached.
Not all the possible forces taken into account have the same computational cost. A peculiar case is given by long-range forces, which require to evaluate the interaction between each pair of particles. In this case, the computational cost of the method increases quadratically with the number of particles considered. Ad-hoc methods are developed to reduce the computational effort, for example combining particles that are far from the particle in consideration and considering them as a single pseudo-particle.

The main disadvantage of DEM is that the maximum number of particles is strongly limited to computational resources. Usually, fluids are considered as made of billions of particles and this results in a huge demand in computational power. Cluster of GPUs can be used for reducing computational time.

Despite this problems, DEM is a powerful resource to simulate a wide variety of granular flows and rock mechanics problems and it can be the unique way to study micro dynamics of systems in which measurements are nearly impossible due to the small scale.

4.1 Discrete Element Method

The Discrete Element Method (DEM) is a Lagrangian method used for calculating the dynamics of large granular systems. In this presentation we use results presented in [4] and [8]. The particle flow is resolved at the particle level. In fact, as described above, DEM calculates the trajectory of each particle considering the influences by other particles, walls or other problem-specific forces. The motion of a particle consists of a rotational and a translational component, therefore the equations that describe the method are the Newton's laws for translations and rotations:

$$\begin{aligned} m_i \frac{du_i}{dt} &= F_i, \\ I_i \frac{d\omega_i}{dt} &= T_i, \end{aligned} \tag{4.1}$$

where m_i is the mass of the particle i, F_i is the force applied to the particle and u_i is the velocity, which is unknown; I_i is the inertia tensor, T_i is the torque applied to the particle and ω_i is the angular velocity, which is also an unknown. The force F_i has to be modeled in order to describe the particle/particle interactions. Usually, as described in [8], F_i takes into account:

- a gravitational component $m_i g$
- particle-particle collisions $\sum_{N_p} F_{i,p}$
- particle-wall interactions $\sum_{N_w} F_{i,w}$
- cohesive interactions $\sum_{N_p} F_{i,c}$

where N_p is the number of the particles in the system and N_w is the number of the walls. We have to mention that other problem-specific forces can be considered, like electromagnetic or chemical contributions. Using the aforementioned forces, the force F_i on the particle is given by:

$$F_i = m_i g + \sum_{N_p} F_{i,p} + \sum_{N_w} F_{i,w} + \sum_{N_p} F_{i,c} \tag{4.2}$$

In this case, Netwton's laws (4.1) take the form:

$$\begin{aligned} m_i \frac{du_i}{dt} &= m_i g + \sum_{N_p} F_{i,p} + \sum_{N_w} F_{i,w} + \sum_{N_p} F_{i,c} \\ I_i \frac{d\omega_i}{dt} &= T_i \end{aligned} \tag{4.3}$$

After having derived the equation (4.3) it is straightforward to notice that the next step is to model the forces $F_{i,p}$, $F_{i,w}$, i.e. the contributions of the interaction between particle i and all other particle and the walls, respectively, and and $F_{i,c}$, i.e. the contribution of the cohesive forces.

For the sake of simplicity, in the development of DEM each particle is assumed to be a sphere. This will impose a limitation on the accuracy of the method, but non-spherical particle can be approximated by several spheres glued together.
The assumption is a necessary simplification, since it allows to develop easily contact and cohesive models and to consider the arising torque T_i as generated exclusively by tangential component of the force F_i.

Particle-particle interactions
For the purpose of modeling the interactions that arise form particle-particle collisions, two different approaches can be used: hard sphere model and soft sphere model.

- *Hard sphere approach*
  The particles are impenetrable and the contacts are instantaneous and perfectly rigid. Only binary are considered and long-distance particle forces are neglected. This approach is mainly useful in dilute systems, where the number of binary collisions prevail.

- *Soft sphere approach*
  When solids exert forces on each other, they are subjected to deformation. In this model, deformation is replaced with an overlap between the two particles taken in consideration. The results are more accurate than the hard sphere model, but the computational effort is much bigger.

The Discrete Element Method is based on the soft sphere model.

The main assumption is that particles which are in direct contact with the particle in consideration influence its motion. A collision between elastic particles generates repulsive forces, which will be directly proportional to the deformation, described by the overlap. Since we are considering deformations, we have to notice that deformation of a body implies energy loss, which depends on deformation speed.

A very intuitive mechanical analogy to this process is given by the spring-damper system, where the motion of a body with mass m is described by

$$m\ddot{x} + \eta\dot{x} + kx = 0, \tag{4.4}$$

where η is the damping coefficient of the dash-pot, k is the stiffness of the spring and x is the distance form the equilibrium position. In Figure 4.1 a visualization of a spring-damper system is provided.
In the dash-pot we have loss of kinetic energy, which is in accordance to the hypothesis of energy loss due to deformation. The solution of this linear ODE is parameter-dependent, and can be

over-damped, critically damped or under-damped. Usually the collision of particle gives rise to under-damped solutions.

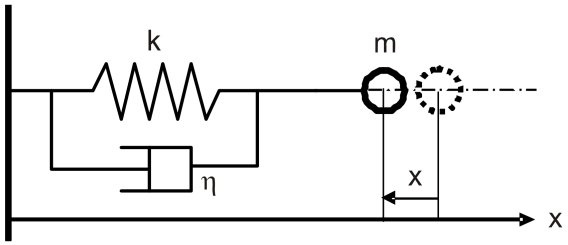

Figure 4.1: **Analogy particle collisions and spring-damper system**.

The analogy between the two phenomena lies in the following processes. Due to external forces, one particle A is pushed towards another B. In the spring-damper system this corresponds to a force that moves the equilibrium position of x. When the point leaves the equilibrium, the spring compresses and a repulsive force is created. This repulsive force corresponds to the reaction force caused by B acting on the particle A. Due to this reaction force, particle A starts to return to its original position after having reached the maximum displacement. Due to energy loss, the velocity of returning will be lower than the velocity before collision.

Using the linear ODE (4.4) not all details of the physical process are described, but we still manage to model a variety of cases, so we will consider the model a good approximation of reality.

The aim is to develop a model for the force $F_{i,p}$, which describes the force applied to particle i due to particle-particle interactions. For the sake of simplicity, we write F_i.
The force F_i will be the sum of all the collisions with the j particles that are in contact with particle i. Therefore we can write

$$F_i = \sum_j F_{ij}. \tag{4.5}$$

We define δ as the overlap that is developed when two particles collide. We consider a collision between particle i and particle j. The overlap δ will have both normal and tangential component δ_n,δ_t. The same applies to the force F_{ij}: we decompose it in normal component F_{nij} and in tangential component F_{tij}. Hence, equation (4.2) becomes:

$$F_i = \sum_j \left(F_{nij} + F_{tij}\right). \tag{4.6}$$

Different equations model these two contributes.
Now we use the analogy between particle collisions and spring-damped systems to build a model for normal and tangential component of the force applied to the particles. We will deal with the stiffness parameter k, the damping coefficient η and the friction coefficient f.

Firstly, we model the normal component of the force, i.e. F_{nij}.
The normal component is given by the sum of the forces due to the spring and the dash-pot. Using the Hertzian contact theory, the force is given by:

$$F_{nij} = (-k_n \delta_n^{3/2} - \eta_{nj}(\mathbf{v}_i - \mathbf{v}_j) \cdot \mathbf{n})\mathbf{n} \tag{4.7}$$

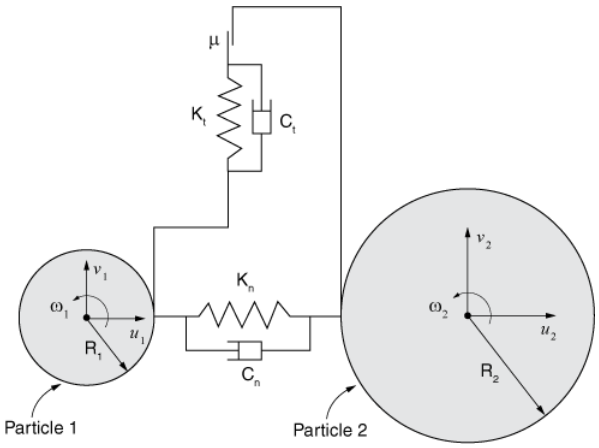

Figure 4.2: **Spring-damper system applied to contact between particles**.

where k_n is the stiffness coefficient in the normal direction, η_{nj} is the damping coefficient in the normal direction, $\mathbf{v}_i$ is the velocity of particle i and $\mathbf{n}$ is the unit vector with direction the line that connects the centers of particles i and j. The power $3/2$ of the displacement may seem strange, but the results are in good accordance with the experimental cases.

Now we model the tangential component of the force, F_{tij}.
For the tangential component we have to distinguish two cases, depending on the ability of the sphere to slide or not. We use a Coulomb-type friction law on the first case, whereas we develop an ad-hoc model for the second. Hence the force is given by:

$$F_{tij} = \begin{cases} -f|F_{nij}|\mathbf{t} & |F_{tij}| \geq f|F_{nij}| \\ -k_t\delta_t - \eta_{tj}\mathbf{V}_{ct} & \text{else} \end{cases} \tag{4.8}$$

where k_t is the stiffness coefficient in the tangential direction, η_{tj} is the damping coefficient in the tangential direction and f is the friction coefficient which is measured empirically, $\mathbf{t}$ is the unit vector in the direction of $\mathbf{V}_{ct}$, which is the slip velocity and is modeled as:

$$\mathbf{V}_{ct} = (\mathbf{v}_i - \mathbf{v}_j) - ((\mathbf{v}_i - \mathbf{v}_j)\cdot\mathbf{n}) + a_i\omega_i \times \mathbf{n} + a_j\omega_j \times \mathbf{n}$$

where a_i and a_j are the radii of the particles i and j.

We will now focus on the modeling for the stiffness coefficients k_n and k_t and the damping coefficients η_n and η_t.

We use Hertzian contact theory for the normal stiffness coefficient k_n. If we know Young's modulus E and Poisson ratios σ for the particles i and j we have:

$$k_n = \frac{4}{3}\left(\frac{1-\sigma_i^2}{E_i} + \frac{1-\sigma_j^2}{E_j}\right)^{-1}\left(\frac{a_i + a_j}{a_i a_j}\right)^{-1/2} \tag{4.9}$$

We use Mindlin's theory for the tangential stiffness coefficient k_t. Knowing the shear modulus H of the two particles i and j we have:

$$k_t = 8\left(\frac{1-\sigma_i^2}{H_i} + \frac{1-\sigma_j^2}{H_j}\right)^{-1}\left(\frac{a_i + a_j}{a_i a_j}\right)^{-1/2}\delta_n^{1/2} \tag{4.10}$$

For the two damping coefficients we use the expression proposed by Cundall and Strack. They calculated the two coefficients as the critical damping conditions for the spring-damped system (4.4), since the bouncing motion after collision should be damped as soon as possible. therefore we get:

$$\eta_n = 2\sqrt{mk_n} \tag{4.11}$$

$$\eta_t = 2\sqrt{mk_t} \tag{4.12}$$

Cohesive Interactions
Granular materials have the ability to resist external tensile stress. This property is caused by microscopic attraction forces between particles, called cohesive interactions, which may have physical and chemical origins. For this part, we follow the dissertations [4] and [14].

The cohesive interaction have the effect to resist to separation, shear or rolling of two particles, restricting the relative particle displacements. Not all the granular materials manifest appreciable effects of the cohesive interactions: it is possible to subdivide the materials in weakly cohesive and strongly cohesive, depending on the influence of the cohesive forces in the macroscopic behavior.
Several physical phenomena can be responsible for the arise of cohesive forces. We will give three examples: electrostatic and Van der Waals forces as *bulk forces* and capillary bridge as *contact force*.

We first introduce the electrostatic case. Electrostatic forces occur if at the particle surface the electrical charges of opposite signs are not balanced. The radial force that arise is this case, between particles i and j is:

$$F_{ij} = \frac{q_i q_j}{4\pi\varepsilon_0\varepsilon_r l^2}, \tag{4.13}$$

where q_i and q_j are the electrical charges of particles i and j, ε_0 and ε_r are the vacuum permittivity and relative permittivity of the medium, respectively and l is the distance between the particle centers.
This force is attractive if q_i and q_j have opposite signs, otherwise it is repulsive.

The Van der Waals forces are inter-particle forces that cause adhesion between particles between each other and the walls. They are particularly strong when smooth surfaces are brought to contact. The Van der Waals force between two spheric particles is described by:

$$F_{ij} = \frac{A}{12l^2}\frac{D_i D_j}{D_i + D_j}, \tag{4.14}$$

where D_i and D_j are the diameters of particles i and j respectively, A is a constant (*Hamaker Constant*) and l is the separation distance at the particles start to interact.

Cohesive forces can arise even when two particles are in contact, developing adhesion. An example is given by capillary bridges. The capillary force is given by:

$$F_{ij} = -2\pi R^*_{ij}\gamma\cos\theta, \tag{4.15}$$

where γ is the liquid surface tension and θ the contact angle and R^*_{ij} is the average of the radii of particles i and j.

Total Interaction
Once we have modeled each component of the forces acting on the particle i, we can go back to (4.3) and rewrite it in a simpler way. We do not consider all the particles, but only those that are in contact with particle i. Hence we do not sum over N_p but over j. Moreover, we sum all the different contributions in the normal direction in F_{nij}^{tot} and in the tangential direction in F_{tij}^{tot}. Therefore we obtain:

$$\begin{aligned} m_i \frac{du_i}{dt} &= m_i g + \sum_j \left(F_{nij}^{tot} + F_{tij}^{tot} \right) \\ I_i \frac{d\omega_i}{dt} &= \sum_j \left(a\mathbf{n} \times F_{tij}^{tot} \right) \end{aligned} \tag{4.16}$$

since the torque T_i is generated exclusively by the tangential component of the forces, as explained in the previous paragraph.

4.2 HADES

HAbaneras Discrete Element Simulator, named HADES, is a discrete element software package, developed by Habanera, that simulates granular flow or mixture problems.
As described in the previous section, using DEM the behavior of the entire material/mixture is simulated by considering contribution of each constituent in the mixture individually. In fact, performing a simulation of the complex interactions between the grains mutually and the influence of the environment on the individual grains, the behavior of the whole mixture can be evaluated.

In HADES the dynamics of the individual particles is evaluated by integrating Newtons second law of motion.
With the technique adopted in the previous section, it is possible to model the total net forces and torques acting on a particle. This is obtained summing the individual forces F_{ij} and torques T_{ij} that act on body i over the number of actuators j, where as actuators we define the processes responsible for the arise of a force.
Knowing the particle state (its position and its velocity) at a particular time, it is possible to obtain the state of particles at a later time integrating the above equations in time. To this purpose various numerical integration schemes can be used. Currently, HADES only supports explicit schemes.

HADES software provide a powerful implementation of the DEM. The outline of the solution procedure is given by:

Outline of solution procedure
Initialize the state of all particles (positions, velocities);
while *(not done)* **do**
  calculate the individual forces that act on the particles at time t (more evaluations at different positions and times may be necessary);
  calculate the net force and torque that act on each body;
  calculate the new velocity and position of each particle at time $t + \Delta t$ by integration over the time period Δt;
  advance the time to $t + \Delta t$;
  advance the particles to the new state;
end

Actuators and Models
Very different processes can be at the origin of forces and each one of them requires careful modeling. As stated before, we call each of these processes an actuator. The result of an active actuator on a particle is a force. For example, gravity, drag and contact are actuators that may contribute to the total net force that acts on a particle.

In HADES the force F_{ij} that acts on a particle i due to an actuator j is independent from the force F_{ik} that acts on the same body but from a different actuator k. Therefore each actuator can evaluate its influence on a group of particles, independently from any other actuators that may be active. For example, the evaluation of the gravity force can be performed independent from the evaluation of the drag force and contact force.

In HADES these actuators are encapsulated in so called Models. Each Model calculates its contribution to the total force that acts on each particle. These Models can be added via the input file.
The user specifies, in an ASCII input file, which models are active during the simulation and what values should be used for the relevant model parameters. For example, if gravity plays no role in the experiment that is being simulated, the user simply does not add this Model. This modular design gives HADES a very strong flexibility and since most of the functionalities are encapsulated in independent Models, it also allows HADES to be extended in a simple way.

Physical objects, such as containers, hoppers, borders and particles, may have arbitrary shapes, but the evaluation of the mutual interactions between arbitrarily shaped objects is in general too computationally expensive. Hence, optimized algorithms are provided for simple shapes, like spheres and planes.

Particular useful Models are those that imply forces on the particles. The implemented Models with this purpose are *collision models*, that calculate the inter-particle contact forces, *drag force models* that calculate the force on a body that moves through a medium and the *gravity model* that calculates the gravitational forces that act on a body.

Other implemented Models control the number of particles that are active during a simulation. To this purpose, a sink model and various generator model are available. The *sink model* deletes from the simulation the particles that enter a user-defined geometrical region, instead other *generator models* are able to generate particles of a user-defined shape and size, at user-defined locations at user-defined times during the simulation.

Property files
In order to run an HADES simulations, user-defined input needs to be considered. The user gives directives through the *property files*. The property file is a text file containing a set of name-value pairs, called properties, that describe the runtime parameters and the models to be used. The general syntax of a property is:

```
name = value;
```

The value of a property can also be a property set, so that it is possible to create a tree-like structure of properties. An example of same parts of a property file is given by the following script:

```
integrator = // define integration scheme and configure integrator
{
    type            = "Heun";
    maxTimeStep     =  5.0e-03;
    minTimeStep     =  5.0e-07;
    fixedTimeStep   =  false;
```

```
};

generator = // define generator type and configure this generator
{
    type               = "ellipsoidGenerator";
    maxElemSize        =      1.0e-02;
    maxElemCurv        =     10.0;
    density            =   1000.0;
    groupName          =  "particles";
    outputFile         =  "myname.out";

generation =
{
        nrOfBatches        = 1;
        bodiesPerBatch     = 8000;
        fireTime           = 0.0;
    };
};
```

Using this kind of syntax for the appropriate properties, the user is able to generate particles with particular shapes, choose which actuators to apply, define the time integration method and all the parameters required for the simulation.

We now list the built-in models, already available in the current version of HADES.

Built-in Models
Models to add bodies to a simulation:

- `ellipseGenerator` generates bodies of elliptical shape
- `ellipsoidGenerator` generates bodies of ellipsoidal shape
- `fromFileGenerator` generates bodies of which the shape description is obtained from file

Model that remove bodies from a simulation:

- `sink` removes bodies from the simulation

Models that apply contact forces and gravity to particles:

- `HertzContact` Hertz Mindlin contact between particles of spherical shape
- `HertzContactPlaneSphere` Hertz Mindlin contact between particles of spherical shape and infinite planes
- `gravity` calculates the gravitational force on bodies
- `segmentContact` contact model between particles of arbitrary shape

Other modulus provide a first (one-way) coupling between particles and fluid. for example the calculation of drag force on particles, but we will discuss them later in the section dedicated to the coupling.

Up to now, HADES provides extensive modulus for contact forces, but cohesive force are missing.

We now give two examples of the usage of HADES software.
In Figure 4.3 the fall of particles in a 2D box is simulated. After the flow is stopped, a circular motion is imposed on a particle and the behavior of all the other particles of the system is explored.
In Figure 4.4 the collision of 3D particles is explored. The particle in the right is provided with a velocity in left direction. All the collisions are accurately caught and in the final time of the simulation all the particles have a non-zero velocity in the left direction. Due to the lack of other collisions, the particle in the left moves away from the system.

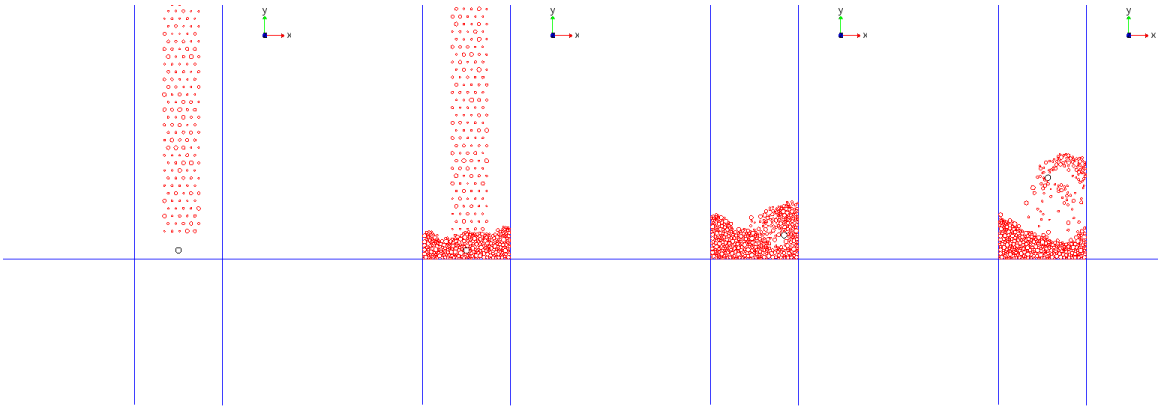

Figure 4.3: **Particles fall and successive perturbation**

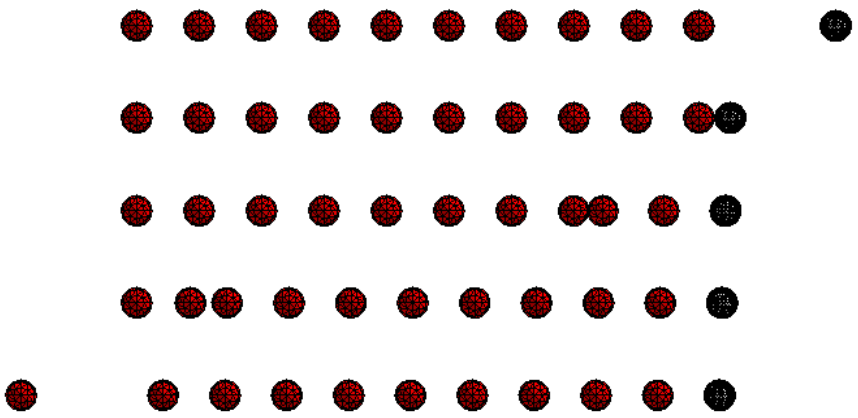

Figure 4.4: **3D Collisions**

4.3 LIGGGHTS

LIGGGHTS is an Open Source Discrete Element Method Particle Simulation Software.

LIGGGHTS stands for LAMMPS improved for general granular and granular heat transfer simulations. LAMMPS is a classical molecular dynamics simulator. It is widely used in the field of Molecular Dynamics. Thanks to physical and algorithmic analogies, LAMMPS is a very good platform for DEM simulations. LAMMPS offers a GRANULAR package to perform these kind of simulations. LIGGGHTS aims to improve those capability with the goal to apply it to

industrial applications. A very detailed documentation is provided in [3].

LIGGGHTS applies the Discrete Element Method described in the previous sections for simulations of interactions of particles between each other and with walls of containers. The equation of motion is numerically integrated, updating the state of each particle every time step. LIGGGHTS is a software written in C++ and it supports parallel computing via MPI. Mainly, it is open-source, but some functionalities are provided only through a commercial license.

The user has to provide the set up of the simulation and the parameters required through a text file, called *input file*. Then, the user will run an executable file giving as input the text file filled with the geometry, material parameters and details of the simulation. It is possible to set the geometry of the problem directly on the input file or importing an user-defined mesh via a STL file.

Input File
In the input file, the user has to set up the domain of the simulation, to choose the models to use (contact, cohesion, ...) and to fix all the parameters necessary for the simulation. The structure is usually given by:

- Definition of the shape of the particle
- Definition of the boundaries of the domain
- Definition of the minimum distance for detecting the neighbors of the particles
- Setting of the physical parameters required by the model used
- Fixing the walls of the simulation or importing STL geometries
- Fixing the region of insertion of the particles
- Choosing if particles with different properties are needed
- Choosing output settings
- Run up to a certain final time

Here we give an example of an input file, which has been used to produce a simulation of a particle drop on a oblique chute. Spherical particles have been used, but with two different sizes: 30% big and 70% small. Result of the simulation at the final time is shown in Figure 4.5.

```
# Simple chute wear test
atom_style    granular
atom_modify   map array
boundary      f f f
newton        off
communicate   single vel yes
units         si

region        domain block -0.5 0.1 -0.2 0.2 -0.4 0.15 units box
create_box    1 domain
neighbor      0.002 bin
neigh_modify  delay 0

# Material properties required for new pair styles
fix  m1 all property/global youngsModulus peratomtype 5.e6
```

```
fix  m2 all property/global poissonsRatio peratomtype 0.45
fix  m3 all property/global coefficientRestitution peratomtypepair 1 0.3
fix  m4 all property/global coefficientFriction peratomtypepair 1 0.5
fix  m5 all property/global k_finnie peratomtypepair 1 1.0

# New pair style
pair_style  gran model hertz tangential history #Hertzian without cohesion
pair_coeff  * *
timestep    0.00001
fix         gravi all gravity 9.81 vector 0.0 0.0 -1.0

# Fix the wall: the chute
fix  cad all mesh/surface/stress file meshes/simple_chute.stl type 1 wear finnie
fix  inface all mesh/surface file meshes/insertion_face.stl type 1
fix  granwalls all wall/gran model hertz tangential history mesh n_meshes 1 &
     meshes cad

# Distributions for insertion
fix  pts1 all particletemplate/sphere 15485863 atom_type 1 density constant 2500
     radius constant 0.0015
fix  pts2 all particletemplate/sphere 15485867 atom_type 1 density constant 2500
     radius constant 0.0025
fix  pdd1 all particledistribution/discrete 32452843  2 pts1 0.3 pts2 0.7

# Region and insertion
group  nve_group region domain
region bc cylinder z 0.0 0.0 0.015 0.05 0.12 units box

# Particle insertion
fix    ins nve_group insert/stream seed 32452867 distributiontemplate pdd1 &
       nparticles 6000 massrate 0.1 insert_every 1000 overlapcheck yes &
       all_in no vel constant 0.0 0.0 -1.0 insertion_face inface

# Apply nve integration to all particles that are inserted as single particles
fix    integr nve_group nve/sphere

# Output settings, include total thermal energy
compute       1 all erotate/sphere
thermo_style  custom step atoms ke c_1 vol
thermo        1000
thermo_modify lost ignore norm no

# Insert the first particles so that dump is not empty
run    1
dump   dmp all custom/vtk 200 post/chute_*.vtk id type type x y z ix iy iz &
       vx vy vz fx fy fz omegax omegay omegaz radius
dump   dumpstress all mesh/gran/VTK 200 post/mesh_*.vtk stress wear cad

# Insert particles
run    100000 upto
unfix  ins
```

In LIGGGHTS it is possible to apply also cohesive models. The simulation of the drop of particles has been performed and in Figure 4.6 and 4.7 final results are given.
As expected, the case with cohesive forces results in the creation of a clump, whereas without cohesive forces a more ordered structure is obtained.

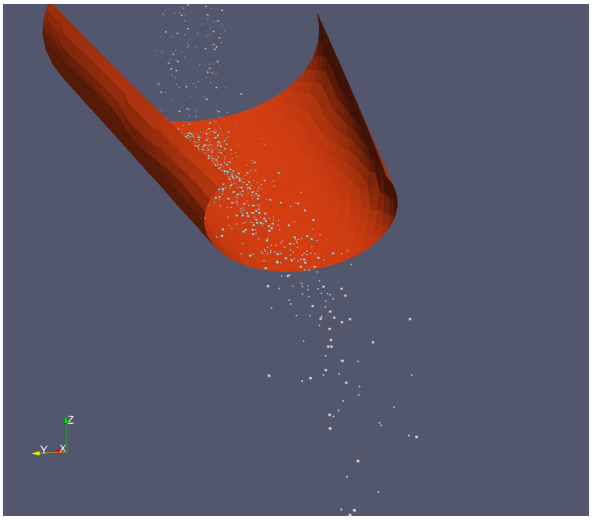

Figure 4.5: **Particle drop on a chute**.

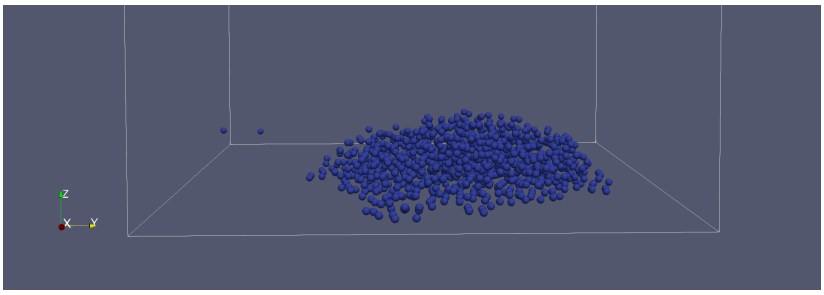

Figure 4.6: **Drop with cohesion**.

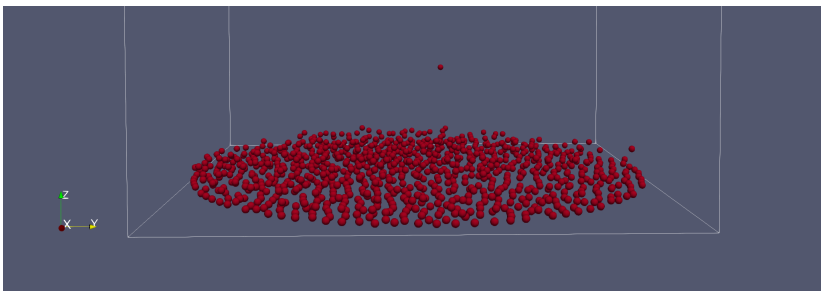

Figure 4.7: **Drop without cohesion**.

Chapter 5

Coupling

Physical simulations often deal with interactions between solid particles and fluids, whereas up to now we considered these two phenomena as independent. In this section, some methods for the interaction between fluids and particles are described and analyzed. This process is called *Coupling*, and we will focus on the coupling between CFD for fluid simulations and DEM for particle interactions.

The first classification of the coupling procedure regards the reciprocal influences between fluid and particles and particle on themselves. In fact, we distinguish 3 different cases:

- *1-way coupling*
  Fluid exerts influence on particle motion but not vice versa. Neglect particles interactions
- *2-way coupling*
  Fluid exerts influence on particle motion and vice versa. Neglect particles interactions
- *4-way coupling*
  Fluid exerts influence on particle motion and vice versa. Resolve particles interactions

Depending to the specific problem, the most efficient coupling can be implemented. The aim of this project is to study the 4-way coupling, therefore we will focus on this case.

In this chapter, we firstly describe the modeling of fluid/particle interactions, describing the different contributions to the phase coupling. Then we discuss some general issues on the coupling procedure, i.e. the time step choice and the contact detection algorithm. Finally, we describe two different approaches that have been developed for the CFD-DEM interactions: *Resolved* and *Unresolved* couplings. Resolved coupling is useful when the size of the particle is bigger than the computational grid used for fluid simulation, whereas unresolved coupling deals with the case of particles which are smaller than the computational grid. We will describe in depth the details of this two distinct methods, for which we follow the references [8] and [10]. Some open source implementations of the coupling procedure are then listed and briefly described.

5.1 Modeling of Fluid/Particle Forces

The fluid dynamics force on a particle can be modeled as the superposition of different contributes. In this section we briefly describe the different components. It is important to notice that not every component will be relevant in every problem: the formulation of fluid/particle

interactions is often problem-specific. In this section, we follow [4] for the description of the different contributions.

Mainly, the influence of fluid on particle motion can be described as the sum of two contributions: drag forces and lift forces.

5.1.1 Drag forces

The drag forces that arise on the particle due to the fluid are:

- Undistrubed flow
- Steady state drag
- Virtual (or added) mass
- Basset term

Undisturbed Flow
This component describes the contribution of the pressure and the shear stress fields in the undisturbed flow, i.e. in the flow without considering the presence of the particle. This term contributes significantly in liquid-particles flows, but it is neglectable in gas-particle flows. The model is described by:

$$F_{ud} = V_d \left(-\frac{\partial p}{\partial x_i} + \frac{\partial \tau_{ik}}{\partial x_k} \right), \tag{5.1}$$

where V_d is the volume of the particle.

Steady state drag
This terms models the drag force that acts on the particle in a velocity field where there is no acceleration of relative velocity between the particle and the conveying fluid. Different models are available in the literature, to describe a variety of situations. We give, as an example, the steady state drag force based on the drag coefficient C_D:

$$F_{ss} = \frac{1}{2}\rho_c A C_D |u - v|(u - v), \tag{5.2}$$

where ρ_c is the density of the fluid carrier, A is the project area of the particle in the direction of relative velocity of fluid and particle and C_D is the drag coefficient. The drag coefficient C_D is strongly influenced by the Reynolds number Re. The dependence of C_D from Re has been extensively studied and there are several models available for different ranges of Re.

Virtual (or apparent) mass effect
The virtual mass effect appears when a body in a fluid is subjected to an acceleration. The fluid is accelerated, and the work necessary to obtain this acceleration is done by the body. Therefore, the global effect is called apparent mass force, since it is equivalent to adding a mass to the sphere.
The virtual mass force can be modeled as:

$$F_{vm} = \frac{1}{2}\rho_c V_d \left(\frac{Du}{Dt} - \frac{dv}{dt} \right), \tag{5.3}$$

being V_d the volume of the particle.
Virtual mass effect appears only in unsteady flows.

Basset Term
This term describes the delay in the development of boundary layers due to the fact that relative velocity changes with time. It describes the viscous effects of an acceleration and it is known in literature also as history term, since its value depends on acceleration history up to the present time. This contribution is modeled as:

$$F_{Bass} = \frac{3}{2} D^2 \sqrt{\pi \rho_c \mu_c} \left[\int_0^t \frac{\frac{d}{dt'}(u_i - v_i)}{\sqrt{t - t'}} dt' + \frac{(u_i - v_i)_0}{\sqrt{t}} \right]. \tag{5.4}$$

We see that also the initial value of the relative velocity influences the final force F_{Bass}.
Basset term appears only in unsteady flows.

5.1.2 Lift forces

The lift forces that arise are:

- Magnus lift
- Saffman lift

Saffman Lift
The Saffman lift arises from the pressure distribution on a particle in a velocity gradient. Let us consider a sphere immersed in a shear flow, with higher velocity at the top of the particle and lower at the bottom. The higher velocity on at the top causes a low pressure, whereas the lower velocity at the bottom gives rise to a high pressure. This differential of pressure develops a lift force.
Let us define the shear Reynolds number Re_G, i.e the Reynolds number based on the velocity difference between the top and the bottom of the particle:

$$Re_G = \frac{D^2}{\nu_c} \frac{du}{dy}, \tag{5.5}$$

where ν_c is the kinematic viscosity of the fluid carrier and D is the diameter of the particle.
If both the relative Reynolds number based on velocity different of fluid and particle Re_r and the shear Reynolds number are small ($\ll 1$) and $Re_r \ll \sqrt{Re_G}$, then the Saffman lift force can be modeled as:

$$F_{Saff} = 1.61 \mu_c D |u_i - v_i| \sqrt{Re_G}. \tag{5.6}$$

Magnus Lift
The Magnus force is the lift generated by the rotation of a particle, due to the presence of a pressure differential between both sides of the particle, caused by the velocity differential due to rotation. The rotation may be caused by other phenomena than velocity gradient.
Generally, the direction of the force is the normal to the plane formed by the rotation vector and the relative velocity vector. If they are orthogonal to each other, the Magnus lift force can be modeled as:

$$F_{Mag} = \frac{1}{2} \rho_c A C_{LR} |v - u| (v - u), \tag{5.7}$$

where A is the project area of the particle and C_{LR} is the lift coefficient due to rotation. The modeling of the coefficient C_{LR} has been explored extensively and a variety of equations are available in the literature.

5.1.3 Total interaction

Exploiting the same procedure adopted by Discrete Element Method in the modeling of particle/particle interactions, the total force is obtained summing all the contributions relevant for the specific problem.
The total force exerted by the fluid on the particle is therefore:

$$F_{fp} = F_{ud} + F_{ss} + F_{vm} + F_{Bass} + F_{Saff} + F_{Mag}. \tag{5.8}$$

Applying third Newton's law we obtain that the force exerted by the particle on the fluid is equal in magnitude and opposite in direction to the total force that the fluid exerts on the particle that we just modeled.

5.2 General issues of Coupling

The coupling between CFD and DEM presents two intrinsic difficulties: an accurate choice for the time step of the simulation and an efficient approach for contact detections.
In fact, for the choice of time steps, we have to consider that it is necessary to catch collision dynamics and satisfy maximum particle overlap constraint (since particles deformation are not modeled directly, but through their overlap).

As far as contact detection is concerned, it is straightforward to notice that a detection of contacts at every time step for every particle is extremely expensive from a computational point of view and, of course, not optimal. The problem is how to detect which particles collide every time step in a computationally efficient way. Various approaches have been proposed and we will discuss the Neighbor List method developed by Verlet in 1967 and the link-cell method proposed by Plimpton in 1995.

5.2.1 Time step choice

It is really important to choose carefully the time steps for the phases of CFD and DEM. Usually, the time step for DEM is smaller than the one for CFD: in most cases the DEM-time step has to be at least an order smaller. In order to compare the two time scales, three important parameters are evaluated:

- CFL number for CFD,
- Rayleigh time for DEM,
- Particle relaxation time for CFD-DEM.

$\boxed{CFD}$
We consider only the presence of a fluid phase. For the numerical integration of the Navier-Stokes equations, the most important parameter to consider is the CFL number (Courant-Friedrichs-Lewy). In an n dimensional case, CFL is defined as:

$$CFL = \Delta t \sum_{i=1}^{n} \frac{u_i}{\Delta x_i}, \tag{5.9}$$

where Δt is the time step and Δx is the space step in the computational grid.
Since the CFL number is a measure of how many cells an infinitesimal volume of fluid passes in

one time step, this has to be smaller than one in order to preserve the stability of the numerical scheme. Therefore, we get a constraint on the CFL which translates in a constraint on the time step Δt:

$$CFL = \Delta t \sum_{i=1}^{n} \frac{u_i}{\Delta x_i} < 1 \quad \Rightarrow \quad \Delta t < 1/\sum_{i=1}^{n} \frac{u_i}{\Delta x_i}. \tag{5.10}$$

DEM

Considering only the interactions between particles, we are in the field of granular flows.
In high density particle regions, the motion of particles is affected not only by forces and torques arising from collisions with particles in the immediate neighbor, but also by disturbances propagating from more distant particles. The propagation of these disturbances is modeled via Rayleigh waves, i.e. surface waves that travel (with both longitudinal and transverse components) near the surface of solids. To ensure realistic force transmission rates and to prevent numerical instabilities, an upper bound for the simulation time step is therefore necessary.

The idea proposed in [13] and [1] is that time step for detecting collision between a particle and its neighborhood should be less than the time it takes for the Rayleigh wave to transverse the minimum size particle in the assembly. The Rayleigh time step proposed is therefore:

$$T_R = \pi r \frac{\sqrt{\rho/G}}{0.1631\nu + 0.8766}, \tag{5.11}$$

where r and ρ are the radius and the density of the particle, G is the particle shear modulus and ν is Poissons ratio. We see that the time-step is material dependent through G. Hence, if we want to model the motion of particles of different materials, we necessarily have to consider the minimum of the different Rayleigh time steps.
Therefore, to prevent numerical instabilities and unphysical results, $\Delta t_{DEM} < T_R$. Often, a fraction of T_R is used for the integration time-step.

CFD-DEM: Particle Relaxation Time

Exploring the interaction between fluid phase and solid particles, the concept of particle relaxation time τ is introduced as a measure of the resistance of a particle to adapt to flow motion: the larger τ, the stronger the resistance. The definition of the particle relaxation time is not unique: depending on the specific problem, different models can be used.
In [4] particle relaxation time τ is modeled, in the Stokes regime ($Re \ll 1$), as:

$$\tau = \frac{\rho_p d^2}{18\mu}, \tag{5.12}$$

whereas in [8] another model is used:

$$\tau = \frac{\rho_p d^2}{18\mu}(1 + 0.15Re^{0.687})^{-1}, \tag{5.13}$$

In order to achieve stability of the numerical method, the DEM time step width has to be lower than particle relaxation time:

$$\Delta t < \tau \quad \Rightarrow \quad \Delta t < \frac{\rho_p d^2}{18\mu}(1 + 0.15Re^{0.687})^{-1}. \tag{5.14}$$

As we see, we have three constraints for two parameters (Δt for CFD and DEM). As a rule of thumb DEM-Δt usually at least one order smaller than CFD-Δt, but mainly the approach is problem-dependent and literature material does not provide satisfactory results.

5.2.2 Contact Detection Algorithm

In order to model accurately the dynamics of contacts, it is important to verify if two particles are colliding, at each time step of the simulation. As stated in the introductory paragraph, it is absolutely not efficient to check at every time step if each particle is colliding with every other particle in the system. This operation would cost n^2 checks, being n the number of particles. Since the check should be done at every time step of the simulation, the process would become too computationally expensive.

We now discuss an approach to overcome to this limitation: the Neighbor List method, proposed by Verlet. The main idea is the periodic construction of a list of potential contacts, in order to exclude *a priori* evaluations of contacts between particles too distant. Every time step, the algorithm checks the list for each particle and evaluates if that particle is colliding with the particles in its neighbor list. Of course, the list is built with a period larger than a simulation time step: we define N as the number of time steps after which the list is updated. In the hypothesis of spherical particles, we include a pair of particles (i and j) if it holds:

$$\|\mathbf{x}_i - \mathbf{x}_j\| \leq r_i + r_j + s, \tag{5.15}$$

where r_i and r_j are the radii of particles i and j respectively and s is the Verlet parameter, that can be chosen between some bounds. If we assume a constant time steps Δt and a maximum particle velocity v_{max}, the number of time steps after which we update the list can be modeled as:

$$N = \frac{s}{2v_{max}\Delta t}. \tag{5.16}$$

Alternatively, another method to build the neighbor list is called link-cell method and it is based on a binning approach on a grid decomposition. In this case, the parameter to choose is the length scale of the binning. A visualization of the two different method is provided in Figure 5.2.2, where only one particle has been considered and not a pair as in (5.15) for Verlet method.

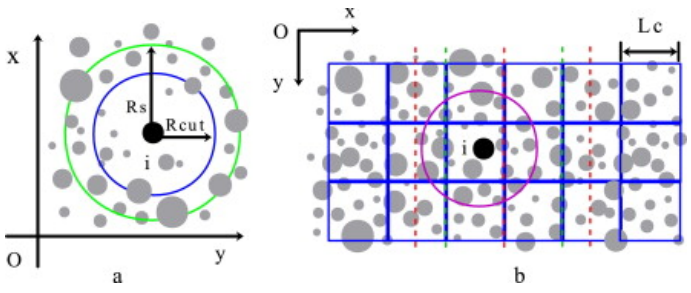

Figure 5.1: **Contact Detection Methods.** The case of a single particle is analyzed. In the left, the scheme of the Verlet algorithm is given and in this case R_{cut} is the figure is the radius of the particle ans R_s is the Verlet parameter s. In the right, the link-cell method is represented.

Comparison between the two approaches is explored in [12], but results are often problem dependent, since lots of parameters require careful tuning. The general trend is that in case of a relatively small number of particles is relatively small, the Verlet algorithm is faster than linked-cell algorithm, and the linked-cell algorithm seems more efficient when the number of particles is large.

5.3 Resolved CFD-DEM

In the resolved coupling we deal with particles which cover multiple cells of the CFD computational grid. The core idea of this method is to add a force term to the Navier-Stokes equations, in order to take under consideration the presence of the solid particles.

Since the particles are larger than the CFD computational grid, it is not possible to consider the presence of the particle in only one CFD cell (e.g. the cell where the centroid of the particle lies): this would lead to unphysical results, in which the fluid would flow even in cells occupied entirely by the solid particle. To overcome this problem, ad hoc methods were developed in order to be able to resolve the fluid in an accurate way. These methods are known in literature as Immersed Boundary Methods (IB) and Fictious Domain Methods (FD).

Fictious domain methods are general techniques developed for solving differential equations on complicated domains. The problem is translated into a simpler domain, solved and then the solution is corrected to satisfy the original problem. Instead, an Immersed Boundary Method is a specific approach for CFD to simulate fluid-structure interactions. Basically, Immersed Boundary Method belongs to Fictuois Domain approaches. We will give two applications of these approaches to Resolved coupling, a fictious domain method presented in [8] and the PISO Immersed Boundary scheme developed in [2].

A Fictious domain Method
In this application of the Fictious domain approach the aim is to perform a correction of the velocity field of the fluid. This can be proved to be equivalent to adding a force term to the Navier-Stokes equations. The approach that we will describe provides satisfactory results for moderate Reynolds number.
We consider only one velocity field and one pressure field in the domain and they are shared by the fluid and the solid. The domain taken into account is provided in Figure 5.2.

Firstly, only the fluid is considered and in the whole domain the velocity field is calculated from the following equations:

$$\begin{cases} \rho\left(\dfrac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\cdot\nabla\mathbf{u}\right) &= -\nabla p + \mu\Delta\mathbf{u} + \mathbf{f} \quad \text{in } \Omega \\ \nabla\cdot\mathbf{u} &= 0 \quad \text{in } \Omega \\ \mathbf{u} &= \mathbf{u}_\Gamma \quad \text{on } \Gamma \\ \mathbf{u}(x, t=0) &= \mathbf{u}_0(x) \quad \text{in } \Omega \\ \mathbf{u} &= \mathbf{u}_i \quad \text{on } \Gamma_s \\ \boldsymbol{\sigma}\cdot\hat{\mathbf{n}} &= \mathbf{t}_\Gamma \quad \text{on } \Gamma_s. \end{cases} \tag{5.17}$$

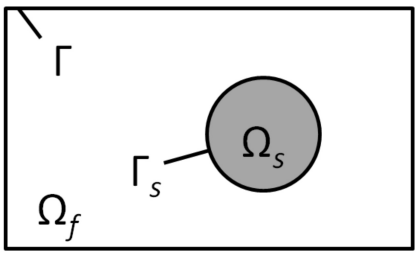

Figure 5.2: **Domain for Resolved Coupling**

The first two equations are the standard Navier-Stokes equations for incompressible fluids. The third and the fourth ones are the boundary conditions on the entire domain and the initial conditions, respectively. The last two equations concern the actual coupling between the fluid and the solid phases: they provide the continuity of velocity field and the normal component of the stress tensor.

Once the data from DEM have been evaluated through a numerical integration, the method consists in the following four phases:

1. Evaluation of an interim velocity field $\hat{\mathbf{u}}$ and its associated pressure field
2. Creation of a new velocity field $\tilde{\mathbf{u}}$
3. Correction of the new velocity field into the final one $\mathbf{u}$
4. Correction of the pressure field

1. Interim velocity field The interim velocity is obtained solving the Navier-Stokes equation in the whole domain, usually using a finite volume method with the PISO (Pressure-Implicit with Splitting of Operators) algorithm.

2. New velocity field The interim velocity is corrected in the particle areas imposing the velocity obtained from the DEM calculations.
This step is equivalent to adding a force term $\mathbf{f}$ to the Navier-Stokes equation, where $\mathbf{f}$ satisfies:

$$\mathbf{f} = \rho \frac{\partial}{\partial t}(\tilde{\mathbf{u}} - \hat{\mathbf{u}}). \tag{5.18}$$

3. Final velocity field Unfortunalety, the new velocity field does not satisfy the divergence-free constraint of the Navier-Stokes equation, which comes from mass conservation under the hypothesis of incompressibility. Therefore, the application of a correction operator is necessary. We define a corrected field $\mathbf{u}$

$$\mathbf{u} = \tilde{\mathbf{u}} - \nabla\phi, \tag{5.19}$$

where ϕ is an unknown scalar field and $\mathbf{u}$ is forced to satisfy the divergence-free constraint. Hence, applying the divergence operator to (5.19) we get a Poisson equation for ϕ:

$$\Delta\phi = \nabla \cdot \tilde{\mathbf{u}}. \tag{5.20}$$

4. Correction of pressure field After having solved (5.20), the pressure obtained from the first step can be corrected in the final one.

Hence, the global outline of the Fictious Domain Resolved method is:

Outline of Fictious Domain Resolved CFD-DEM coupling
while *(not done)* **do**
DEM solver: evaluation of positions and velocities of particles)
Data from DEM solver are passed to CFD solver
Evaluation of interim velocity field
Particle tracking: locate cells occupied by each particle
Correction of velocity in the cells occupied by particles
Evaluation of fluid forces acting on particles
Data from CFD solver are passed to DEM solver for the next time step
Divergence-free correction of velocity field
Evaluation of other equations (i.e. concentrations, ...)
end

Immersed Boundary Method
The fictious domain approach presented in the last paragraph gives accurate results at moderate Reynolds numbers. According to [7], at low Reynolds numbers the results are not satisfactory anymore, so in 2015 another approach was proposed. This method is known as PISO Immersed Boundary and it was presented in [2]. It is a slightly more complicated method, since it is based not only on the correction of velocity and pressure fields, but also of the force term. The results are valid also in the case of low Reynolds numbers.

The idea of the PISO-IB scheme is to add an immersed boundary method to the standard PISO scheme. This is pursued using the PISO loops to impose the velocity of the immersed rigid body while maintaining mass conservation. In every iteration a continuous forcing term is updated and added to Navier-Stokes equation to take into account the immersed body and its motion. The method can be implemented and applied in parallel and it can be used with unstructured polyhedral meshes.

The flow of the algorithm is very similar the the approach developed using the Fictious domain Method. The main improvement is a correction of the forcing term which is performed at the end of every the PISO loop. The main structure of the algorithm is:

1. Detection of immersed body through cell and vertex flagging
   A solid fraction β_i is generated for every cell i
2. Evaluation of forcing term, based on previous data
3. Evaluation of interim velocity from a momentum predictor
4. Start of PISO loop
   - Correction of velocity
   - Solution of pressure correction equation
   - New correction of velocity
   - Correction of pressure
   - Correction of forcing term
     This is the main feature of the method. Forcing term is corrected using the difference between current velocity and the the velocity prescribed within the immersed body.

$$f_i^{new} = f_i^{old} + \frac{\alpha\beta}{\Delta t}(u_{i,ib} - u_{i,current}), \tag{5.21}$$

where β_i is the solid fraction evaluated in step 1. and $\alpha \in]0, 0.9]$ is a relaxation parameter

General issues of Resolved Coupling
Resolved method belongs to the class of Direct Numerical Simulations (DNS), therefore, in order to have very precise results, a high resolution of the fluid mesh in the area of the particles is required. This constraint leads to enormous computational costs, even for small problems. Some remedies can be applied to overcome this limitation.

The first improvement is given by *dynamic local mesh refinement*. This process consists in a mesh refinement around the particles. When a particle moves on, the cells are coarsened again. Especially for dilute particle systems this has a large effect. This feature is already provided by OpenFOAM itself.

Another improvement can be *parallelization*, but in this case, particular attention is to be given in the communication between processors. A void fraction distribution model has been developed to take care of this issue.

5.4 Unresolved CFD-DEM

Unresolved CFD-DEM coupling deals with particle with sizes smaller than the CFD computational grid.
Since different number of particles (of different sizes) can occupy one CFD grid cell, it is useful to introduce a new variable α, which represents the volume fraction occupied by the fluid in a cell. Therefore, we use the so-called locally averaged Navier-Stokes equations for the unknown $\mathbf{u}$, velocity of the fluid phase:

$$\begin{cases} \rho\left(\dfrac{\partial(\alpha\mathbf{u})}{\partial t} + \nabla\cdot(\alpha\mathbf{u}\mathbf{u})\right) &= -\alpha\nabla p + \mu\Delta\mathbf{u} + R_{pf} \\ \dfrac{\partial\alpha}{\partial t} + \nabla\cdot(\alpha\mathbf{u}) &= 0. \end{cases} \tag{5.22}$$

where R_{pf} is the force exchange term, i.e. it takes into account the interaction between the fluid and particle phases. It is evaluated as:

$$R_{pf} = K_{pf}(\mathbf{u} - \mathbf{u}_p), \tag{5.23}$$

where $\mathbf{u}_p$ is the velocity of the particle (taken from DEM data) and K_{pf} is a coefficient. The three major contributions on the interaction between fluid and particles are the gradient of pressure, viscous and drag force. Since the pressure gradient and the viscous term are already taken under consideration in the stress tensor, we model the coefficient K_{pf} using only the contribution of drag forces. Being V the volume of the cell, we define $f_{d,i}$ as the drag force acting on the fluid due to particle i and we get:

$$K_{pf} = \frac{\sum_i f_{d,i}}{V}. \tag{5.24}$$

Other forces may be relevant depending on the specific problem and they can be simply added in the expression of K_{pf}. For example, Magnus force for the rotation of particles, virtual mass force for particle acceleration, Saffman force for gradient of fluid velocity leading to shear. The modular feature of the coupling allows to consider these contributions, properly modeled, directly in the exchange term R_{pf} through K_{pf}.

As stated before, usually only drag forces are relevant and various techniques can be applied in the modeling of K_{pf}. One possibility has been presented in section 5.1, but other models have been developed.
In fact, we now consider a combination of Wen and Yu model (for $\alpha > 0.8$) and Ergun model (for $\alpha \leq 0.8$), which we present in the following paragraph, but it is important to notice that the choice of the model is not unique, since several models are available. We define d as the diameter of the particle under consideration, the Reynolds number base on relative velocity Re_p and the drag coefficient C_d as:

$$Re_p = \frac{|\mathbf{u} - \mathbf{u}_p|\rho d}{\mu}, \tag{5.25}$$

$$C_d = \frac{24}{\alpha Re_p}[1 + 0.15(\alpha Re_p)^{0.687}]. \tag{5.26}$$

Hence, K_{pf} is modeled as:

$$K_{pf} = \begin{cases} C_d \frac{3}{4} \frac{\alpha(1-\alpha)|\mathbf{u} - \mathbf{u}_p|}{d} \alpha^{-2.65} & \alpha > 0.8 \\ 150 \frac{(1-\alpha)^2 \mu}{\alpha d^2 \rho} + 1.75 \frac{(1-\alpha)|\mathbf{u} - \mathbf{u}_p|}{d} & \alpha \leq 0.8 \end{cases} \tag{5.27}$$

The DEM equations are exactly those presented in the section concerning the DEM approach and they have to be solved before the CFD phase, in order to add to the locally averaged Navier Stokes equations the proper terms for the specific problem.

The outline of the complete algorithm is given by the following pseudo-code:

Outline of Unresolved CFD-DEM coupling
while *(not done)* **do**
  DEM solver: evaluation of positions and velocities of particles)
  Data from DEM solver are passed to CFD solver
  Particle tracking: for each particle determine the cell in which it lies
  Determine the particle volume fraction and a mean particle velocity for each CFD cell
  Evaluation of fluid forces from particle volume fraction, for each particle
  Evaluation of exchange terms, for each cell
  Evaluation of fluid velocity (considering particle volume fraction and exchange terms), for each cell
  Data from CFD solver are passed to DEM solver for the next time step
  Divergence-free correction of velocity field
  Evaluation of other equations (i.e. concentrations, ...)
end

Coarse Averaging procedures
One of the most challenging steps in this approach is the interpolation of a Lagrangian property, the particles volumes fraction, from the DEM side to an Eulerian property, the volume fraction field, which is defined on the fixed Eulerian grid of the CFD simulation. The same process has to be performed also for particle velocity and fluid-particle interaction force.
Hence we have to consider methods to interpolate the following physical quantities:

1. Solid volume fraction α,
2. Solid phase velocity $\mathbf{u}_p$,

3. Fluidparticle interaction force R_{pf}.

In the literature, this process is often called "coarse graining" or "averaging". We therefore require an interpolation and different approaches have been proposed. A sum up of the different methods is provided in [16] and [15]
The features that an ideal coarse graining procedure should have are:

1. Conserve relevant physical quantities
2. Handle particles both in the interior cells and the cells near boundaries without producing artifacts
3. Achieve relatively mesh-independent results
4. Be convenient for implementation in parallel
5. Produce smooth coarse grained fields even with the presence of a few large particles in relatively small cells

Some approaches developed for this purpose are Particle centroid method (PCM), Divided particle volume method (DPVM), Statistical Kernel method and the Diffusion-based coarse graining. For the sake of simplicity we will describe these methods for the scalar quantity α (solid particle fraction), but the approches can be generalized for the vector fields $\mathbf{u}_p$ and R_{pf}, component-wisely.

- Particle Centroid Method (PCM).
  It consists in summing over all particle volumes in each cell where the particle centroid lies, to obtain cell-based solid volume fraction. This method is easy to implement in CFD solvers, but it can lead to large errors when cell size to particle diameter ratios are small. Unphysical results can be obtained (for example, particle volume fraction greater than 1).

- Divided Particle Volume Method (DPVM),
  The volume of a particle is divided among all cells that it overlaps with, according to the portion of the volume within each cell. Hence, the solid volume fraction in any cell never exceeds 1 and large gradients in the obtained field are prevented. DPVM works for arbitrary meshes, as long as the particle diameter is smaller than CFD cell size.

- Two grid formulation
  Idea is to use two independent meshes for the averaging and for the CFD simulation. The averaging mesh is chosen based on particle diameters to ensure that the cell sizes are larger than particle diameters. The CFD mesh is chosen according to flow resolution requirements.

- Statistical Kernel method
  The volume of each particle is distributed to the entire domain according to a weight function called kernel function $h(\mathbf{x})$. The solid volume fraction at location $\mathbf{x}$ consists of the superposition of the distributed volumes from all particles

- Diffusion-based coarse graining
  Firstly, an initial value $\alpha_0(\mathbf{x})$ is obtained using PCM. Then, a transient diffusion equation for $\alpha(\mathbf{x}, t)$ is solved with initial condition $\alpha_0(\mathbf{x})$ and no-flux boundary conditions. The result, is a field $\alpha(\mathbf{x}, t)$ and it is the solid volume fraction field to be used in the CFD-DEM formulation. A critical parameter is the final time T for the solution of the diffusive equation, which has to be a physical parameter characterizing the length scale of the coarse graining. Diffusion equation is solved on the same mesh as the CFD mesh.

In Figure 5.3, a representation of the main features of the standard techniques for the coarse graining procedure is given.

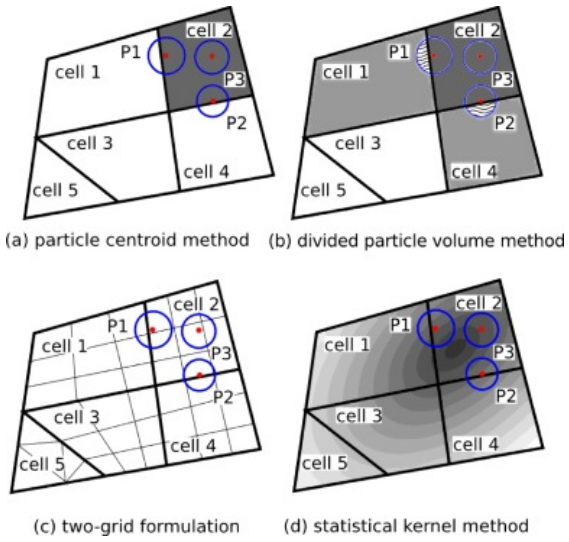

Figure 5.3: **Coarse graining methods**.

We give an example of the diffusion-based coarse averaging procedure for the case of particle volume fraction. As stated before, we first apply the Parcel Centroid Method.
For every cell, we sum up all the particles volume to their host cells, i.e. the cell within which the particle centroid is located. In this way we obtain the total particle volume in each cell.
The solid volume fraction of each cell is then obtained by dividing the total particle volume in the cell by the total volume of the cell. We consider the cell k and therefore the particle volume fraction $\alpha_{s,k}$ is evaluated as:

$$\alpha_{s,k} = \frac{\sum_i^{n_{p,k}} V_{p,i}}{V_k}, \tag{5.28}$$

where V_k is the volume of cell k, $V_{p,i}$ is the volume of particle i contained in cell k and $n_{p,k}$ is the total number of particle whose centroid lies in cell k.
We therefore obtained a cell-based field for the particle volume fraction. We will refer to this field as $\alpha_s^0(\mathbf{x})$ and this will be the initial condition for the diffusion equation that is at the core of the method.

Usually, the initial condition $\alpha_s^0(\mathbf{x})$ is characterized by the presence of steep gradients in the field. In fact, the entire volume of a particle is added to the cell where the centroid lies, even if the centroid is near the border of the cell and therefore the actual volume would occupy different cells. The diffusion-based procedure is able to stabilize the result, since the diffusion equation essentially redistributes particle volumes within the field conserving automatically total solid volume in the domain through the diffusion.

We now give the Cauchy problem that is to solve to apply this method:

$$\begin{cases} \dfrac{\partial \alpha_s}{\partial \tau} - \Delta \alpha_s &= 0 \\ \alpha_s(\mathbf{x}, 0) &= \alpha_s^0(\mathbf{x}) \qquad \leftarrow \text{from PCM} \end{cases} \tag{5.29}$$

It is important to notice that no-flux conditions (i.e. homogeneous Neumann conditions) have

Figure 5.4: **Mesh and particle distribution.** In the right, the mesh and particle distributions are given for the case of two grid method is given. In the left, they are given for all the other approaches.

to be imposed to ensure mass conservation.

Once we have modeled the procedure, we have to solve the diffusion equation numerically. The discretization of the equation exploit the already-built CFD mesh, but it remains to fix how many time steps are necessary to obtained satisfying results in the diffusive process. It has been shown that a final time T equal to one or three time steps τ is enough. In order to estimate the time step τ, we exploit the equivalence between the diffusion-based coarse graining and a Statistical Kernel method based on the Gaussian distribution.
It is possible to prove, in fact, that the diffusion-based procedure is equivalent to a Gaussian Kernel method if the bandwidth of the Gaussian distribution b satisfies:

$$b = \sqrt{4\tau}, \tag{5.30}$$

where τ is the time step of the diffusion procedure.
Imposing a value for the Gaussian bandwidth b, (typically in the literature $b = 6d$, being d the diameter of the particle), we obtain a value for the diffusion time step τ. Solving numerically the diffusive equation (5.29) for just 1 to 3 time steps τ, we obtain satisfactory results for the particle volume fraction field $\alpha_s(\mathbf{x})$.

In Figure 5.4 we specify the mesh and particle distribution for an experiment on the validity of the various approaches described above. In Figure 5.5 the results of the coarse graining procedure are given. It is straightforward to notice that the diffusion-based procedure shows smooth results, without steep gradients in the particle volume fraction field.

The same procedure can be implemented to interpolate vector quantities. In that case the diffusion equation is solved component-wisely.

Mesh convergence results and in-depth comparison of the performance of these approaches are explored in [16]. The diffusion method proposed by the authors manages to achieve very good results in both mesh-convergence and simplicity of implementation. Hence, an implementation of the coupling between CFD and DEM should be based on the diffusion coarse graining approach.

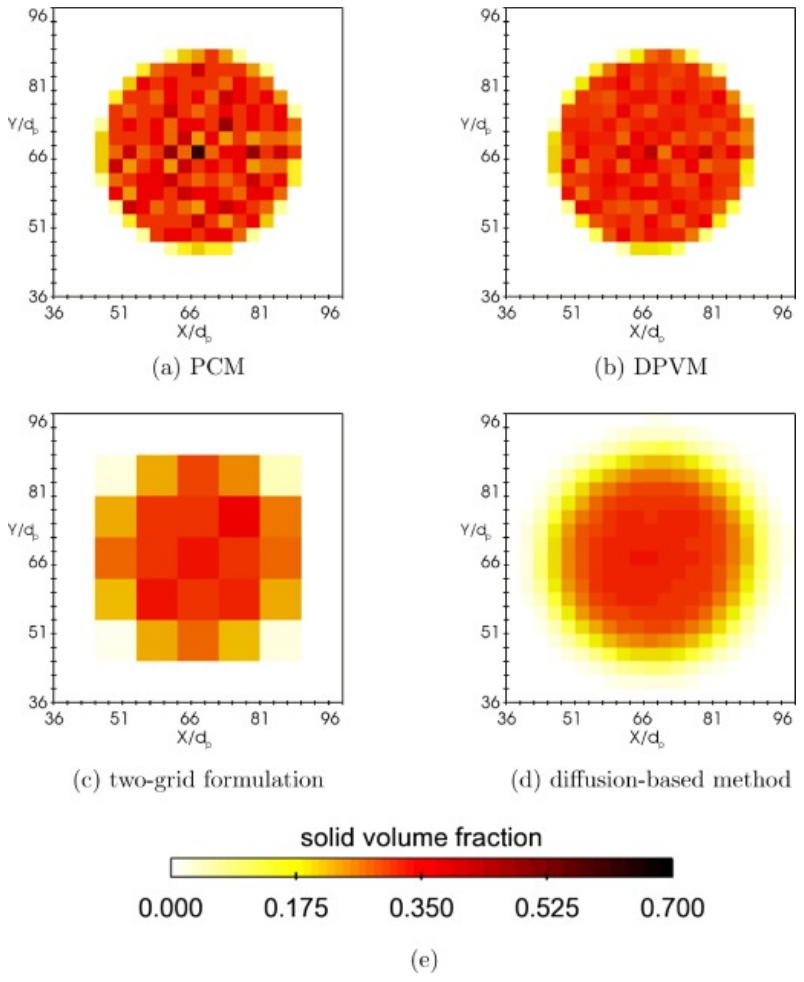

Figure 5.5: **Results of coarse graining procedure.** The different coarse graining methods are applied to the cases described in Figure 5.4.

5.5 Resolved/Unresolved Coupling

Adopting one of the two techniques presented in the previous sections is not enough to simulate every physical process. It is of course possible to have situations in which both large and small solid particles are present and important. Therefore, one of the current research areas is to develop methods and algorithms in which both the cases are considered and solved. A first attempt to develop a strategy for the solution of the problem has been published in [9], in which an algorithm has been developed for an implementation on ANSYS/Fluent platforms.

The coupling developed is a combination of the resolved and unresolved approaches. The particle-fluid interaction are considered at different scales: cell level for fluid phase and particle level for particulate phase.

The idea is, at each CFD time step, to evaluate firstly the fluid forces acting on small particles using a DEM solver to take into account all the inter-particle forces, the influence of walls and the influence of large objects, using data from the previous CFD time step. These evaluations are done until the time of CFD simulation is reached.
After we have reached the synchronization between CFD time and small-particles DEM time, we can move forward on the algorithm and start the evaluation of large-particles dynamics. The calculation of cell volume fraction and momentum source terms is performed through the evaluations of the influences of fluid flow, small particles and walls on the large particles.
This process is performed until the final time of simulation is reached.

In Figure 5.6 we report a numerical result obtained in [9], to give an idea of the capability of the approach. Particle distribution in a fluidized bed with an immersed free-moving tube at different times is plotted

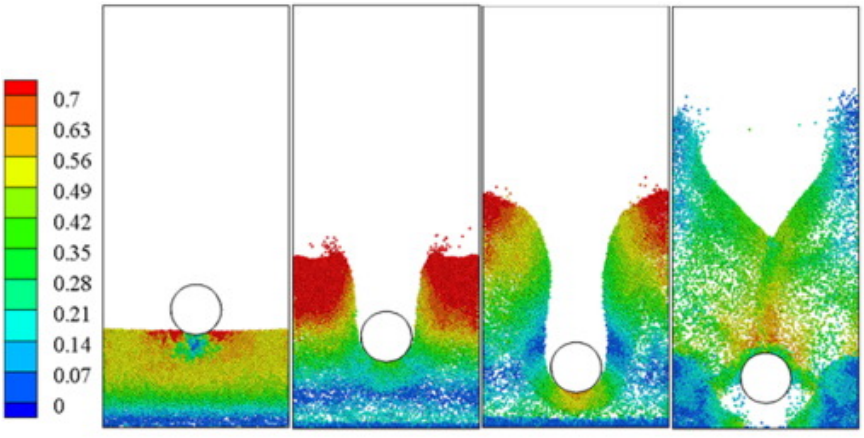

Figure 5.6: **Fluidized bed with an immersed free-moving tube** Results are obtained in [9]. Particle distribution is plotted and particles are colored by velocity magnitude.

5.6 Current implementations

Some libraries are available to simulate the (resolved or unresolved) coupling between fluid and particle simulation.

- `DPMFoam` is a library provided by the standard distribution of OpenFOAM. It does not implement the Discrete Element Method, but DPM (Discrete Parcel Method). The idea is to aggregate clouds of particles and treat them as one big computational particle. This method has strong limitations: the dynamic properties (size, velocity, restitution coefficient, density, ...) for each particle in the parcel have to be the same and the dynamic is solved only at cloud level, not at particle level.

- `sediFoam` is a library developed by Sun and Xiao (2015), for CFD-DEM unresolved coupling in OpenFOAM. The implementation has been described in [17]. The main focus of this library is the simulation of sediment transport and fluidized beds. The most peculiar feature provided in this coupling is the diffusion-based coarse graining approach to perform the evaluation of particle volume fraction, as described in the previous section. The method seems stable for particles sizes at most two times the CFD cell size. The library couples OpenFOAM 2.4.0 with LAMMPS 1-Fed-2014.

- CFDEM is a set of libraries which probably provide the most complete implementation of the coupling. The software is provided by DCS Computing in open source and commerical version. It has been developed both for resolved and unresolved coupling. The solver which has been tested in the outlook of this work is `cfddemsolverIB`. In the resolved approach the Fictious Domain Method described in section 5.3 has been implemented. In the unresolved approach a standar technique has been considered. It couples OpenFOAM-5.x and LIGGGHTS-3.8.0

- `coupledPimpleFoam` is a library developed by Dynaflow Research Group as an expansion of the standard `PimpleFoam` library provided by OpenFOAM. It features a simple adding term to the Navier Stokes equation to take into account forces arising from the presence of particles in the flow. This method can be considered as unresolved. Some severe limitations of this library are:

  1. DEM update is performed in every CFD step: this causes constraints on the CFD time step, since DEM time step usually has to be very small
  2. Particle fraction field α is not evaluated. Various drag models are based on this quantity, therefore they cannot be considered
  3. The coarse graining procedure is intrinsically PCM, therefore huge gradients can be present in the forcing term in the momentum equation.

Chapter 6

Conclusion

6.1 Directions of future work

The field of numerical simulations of fluid/particle system is relatively new and therefore improvements are developed continuously, mostly in the latest years. New methods are proposed and implemented every months, making this topic particularly thriving and inspiring.
Some of the possible future directions for the development of the MSc Thesis Projects are the following:

- Improvement of the in-house developed solver `pimpleCoupledFoam`, implementing a diffusion-based coarse graining procedure for the unresolved case and implementing from scratch an immersed boundary method (fictious domain method or PISOIB) for the resolved case. Some problems of the current implementations are:

  1. DEM update is performed at every CFD step. This is not efficients since the DEM time step is required to be very small and therefore this constraint is imposed also at CFD level
  2. PCM is adopted as interpolation procedure for momentum exchange term. This can cause huge gradients in the exchange terms between neighbor CFD cells
  3. The particle volume fraction α_s is not calculated. This imposes a restriction on drag model to be used (simple sphere drag has to be used and more advanced model have to be discarded)

- Implementation of a coupling from scratch between OpenFOAM and HADES. This path would require extensive coding and a preliminary study on the actual feasibility of the extensions of the various models.

- More theoretical research on optimal time steps for the CFD, DEM and Coupling in order to obtain satisfactory physical result minimizing the computational costs.

Bibliography

[1] M. Afkhami et al. "Fully coupled LES-DEM of particle interaction and agglomeration in a turbulent channel flow". In: *Computers & Chemical Engineering* 78 (2015), pp. 24 –38. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2015.04.003.

[2] Bruno Blais et al. "A semi-implicit immersed boundary method and its application to viscous mixing". In: *Computers & Chemical Engineering* 85 (2016), pp. 136 –146. ISSN: 0098-1354. DOI: https://doi.org/10.1016/j.compchemeng.2015.10.019.

[3] DCS Computing. "LIGGGHTS(R)-PUBLIC Documentation, Version 3.X". In: (2018). URL: http://liggghts.com.

[4] C.T Crowe et al. *Multiphase Flows with Droplets and Particles*. Second Edition. CRC Press, 2011.

[5] J. H. Ferziger and M. Peric. *Computational Methods for Fluid Dynamics*. First Edition. Springer, 2002.

[6] The OpenFOAM Foundation. "OpenFOAM User Guide Version 5.0". In: (2017). URL: http://openfoam.org.

[7] C. Goniva et al. "Open source CFD-DEM modeling for particle-based processes". In: *Proceedings of Eleventh International Conference on CFD in the Minerals and Process Industries*. Dec. 2015.

[8] A. Hager. "CFD-DEM on Multiple Scales - An Extensive Investigation of Particle-Fluid Interactions". PhD thesis. Johannes Kepler Universitt Linz, 2011.

[9] Yi. He, Andrew E. Bayly, and Ali Hassanpour. "Coupling CFD-DEM with dynamic meshing: A new approach for fluid-structure interaction in particle-fluid flows". In: *Powder Technology* 325 (2018), pp. 620 –631. ISSN: 0032-5910. DOI: https://doi.org/10.1016/j.powtec.2017.11.045.

[10] C. Kloss et al. "Model, algorithm and validation for opensource DEM and CFD-DEM". In: *Progress in Computational Fluid Dynamics* 12.2-3 (2012), pp. 140–152. DOI: https://doi.org/10.1504/PCFD.2012.047457.

[11] P. K. Kundu, I. M. Cohen, and D. R. Dowling. *Fluid Mechanics*. Sixth Edition. Academic Press, 2015.

[12] Wan-Qing Li et al. "Comparison research on the neighbor list algorithms: Verlet table and linked-cell". In: *Computer Physics Communications* 181.10 (2010), pp. 1682 –1686. ISSN: 0010-4655. DOI: https://doi.org/10.1016/j.cpc.2010.06.005.

[13] Zemin Ning and Mojtaba Ghadiri. "Distinct element analysis of attrition of granular solids under shear deformation". In: *Chemical Engineering Science* 61.18 (2006), pp. 5991 –6001. ISSN: 0009-2509. DOI: https://doi.org/10.1016/j.ces.2006.03.056.

[14] F. Radjai and F. Dubois. *Discrete-element Modeling of Granular Materials*. First Edition. Wiley, 2011.

[15] Rui Sun and Heng Xiao. "Diffusion-based coarse graining in hybrid continuum-discrete solvers: Applications in CFD-DEM". In: *International Journal of Multiphase Flow* 72 (2015), pp. 233 –247. ISSN: 0301-9322. DOI: https://doi.org/10.1016/j.ijmultiphaseflow.2015.02.014.

[16] Rui Sun and Heng Xiao. "Diffusion-based coarse graining in hybrid continuum-discrete solvers: Theoretical formulation and a priori tests". In: *International Journal of Multiphase Flow* 77 (2015), pp. 142 –157. ISSN: 0301-9322. DOI: https://doi.org/10.1016/j.ijmultiphaseflow.2015.08.014.

[17] Rui Sun and Heng Xiao. "SediFoam: A general-purpose, open-source CFD-DEM solver for particle-laden flow with emphasis on sediment transport". In: *Computers & Geosciences* 89 (2016), pp. 207 –219. ISSN: 0098-3004. DOI: https://doi.org/10.1016/j.cageo.2016.01.011.

[18] P. Wesseling. *Principles of Computational Fluid Dynamics*. First Edition. Springer, 2001.