Laboratory of Multiscale Modeling and Simulation

Thesis for the Master Degree in M2 Analysis, Modeling, and Simulation

Simulations of triboelectric charging in gas-solid fluidized beds with polydisperse particles

Author:
Josue Murillo-Tobar

Supervisor:
PhD. Amine Chadil

Marne-la-Vallée, September 6, 2024

Acknowledgements

My sincere gratitude goes to my professors at Université Paris-Saclay for providing me with an excellent education. I am particularly grateful to Professor Christophe Chalons, one of the coordinators of the program, for his constant support and dedication throughout the master's program.

My deepest appreciation goes to Diego Chamorro, who has been a significant influence on improving the level of research in my country, Ecuador. His dedication to advancing research has motivated me to keep moving forward and never give up on my goals.

My heartfelt thanks go to my parents and brother for their unwavering support and love. I am also grateful to my friends Adrian, Guido, Alex, Naraya, Danny, David L., David H., Ignacio, Jhon, Isabel, Belen, and Geremy for providing me with encouragement and companionship throughout my journey.

A special thanks goes to Viviana, who has been by my side, supporting me through every step. Thank you, Viviana, for your constant encouragement and for believing in me.

Thank you all for your support and guidance.

Josue Murillo
Marne-la-Vallée, September 2024

Contents

Introduction

This internship project was conducted within the TCM team at the MSME Laboratory, focusing on modeling and simulating fluid mechanics for industrial applications. One key area of application is gas-solid fluidization technology, widely used in industries such as petrochemicals, pharmaceuticals, food processing, and agriculture due to its efficiency in processes like drying, catalytic reactions, and mixing [28, 60, 72]. This efficiency is largely due to enhanced mass and heat transfer resulting from frequent interactions between solid particles and the fluidizing gas. However, a significant challenge arises from the electrostatic charges generated by these interactions, which can disrupt fluidized bed behavior and damage equipment [35]. Addressing these challenges requires advanced numerical modeling techniques to accurately capture the complex dynamics of gas-solid flows.

To better understand electrostatic interactions in fluidized bed operations, multi-scale numerical simulations are crucial. In particular, for modeling charge transfer in insulating particles, particle-resolved Direct Numerical Simulation (PR-DNS) is essential as it allows tracking the charge distribution on the particle surface [80]. A valuable method for PR-DNS simulation is the Viscous Penalty Method (VPM) using a fully coupled solver, which involves penalizing viscosity to ensure the solid-like behavior of particles within the fluid [79]. However, VPM still requires improvements, especially concerning collision handling. One promising solution is to incorporate the Adaptive Mesh Refinement (AMR) method.

The introduction AMR is particularly relevant when simulating the collision effects in gas-solid flows, where localized regions may require finer resolution to accurately capture complex interactions at the microscopic level. AMR dynamically adjusts the grid resolution, focusing computational resources where they are most needed, thus minimizing computational cost without losing accuracy [8, 7].

This report is structured as follows: Chapter 1 provides context for the challenges associated with electrostatic interactions in fluidized beds, introducing essential concepts for modeling these interactions and the multiscale approach vital to this problem. Chapter 2 offers an overview of the key components of the Viscous Penalty Method and the fully coupled solver, highlighting the advantages of this approach in resolving particle-fluid interactions. Finally, Chapter 3 presents the implementation of the Adaptive Mesh Refinement method in our case study, along with the results of implementing the AMR method for the one-dimensional case.

Chapter 1

Electrostatic charging in industry

In various industries such as petrochemicals, food processing, pharmaceuticals, and agriculture, gas-solid fluidization technologies are widely employed for processes like drying, catalytic reactions, and mixing [28, 60, 72]. This broad application is attributed to the high efficiency of mass and heat transfer, facilitated by the frequent interactions between solid particles and the fluidizing gas, particularly through collisions between the fluidized particles and between the particles and the walls of the fluidization column. However, these numerous collisions generate electrostatic charges, which can alter the intended behavior of the fluidized bed, pose risks to both equipment and operators, and lead to the formation of layers on the column walls and other surfaces [35]. This buildup often necessitates frequent shutdowns for cleaning. Additionally, in the pharmaceutical industry, the accumulation of electrostatic charges not only contributes to this sheeting formation but also increases the risk of dust explosions [4].

According to Mehrani et al.,(2017) [59], the polyethylene industry has faced several challenges related to the generation of electrostatic charge. One significant issue is the formation of layers along the reactor walls during the catalytic polymerization of ethylene to produce polyethylene. This problem, known as "sheeting," occurs due to the accumulation of charge within the fluidized bed, which causes catalytic particles to adhere to the walls. As a result, the heat generated from the exothermic polymerization cannot dissipate in these areas, leading to the melting of particles and the formation of layers on the reactor walls. This problem limits the reactor's operation to only a few hours before it requires cleaning. Regarding the research on this phenomenon, Hendrickson (2006) [38] provides a comprehensive literature review on the electrification in polymer reactors that leads to sheeting. He also discusses various studies aimed at understanding the underlying mechanisms behind this issue. However, Hendrickson points out that all these studies were conducted under conditions that do not accurately reflect those in a commercial polymer reactor. The research was performed using small-scale equipment, with air humidity levels and particles whose properties differ significantly from those of polymers like polyethylene. Consequently, Hendrickson concludes that further investigation is needed to better understand the formation of layers in polymer reactors.

In general, the electrification of fluidized beds has been a subject of study for decades [18]. Despite the progress made and the numerous solutions proposed, the problem persists, primarily due to the inherent complexity of both the electrostatic phenomena and the fluidization process. In this chapter, we provide a brief introduction to some basic concepts in electrostatics that are essential for simulating triboelectric phenomena in particle-laden flows. Subsequently, we offer a concise overview of the different approaches for modeling and simulating gas-solid fluidized beds, highlighting their relationship with triboelectric effects.

1.1 Triboelectrification

Triboelectric charging, also known as contact electrification, occurs when two neutral surfaces are rubbed together, causing one surface to become positively charged and the other negatively charged, especially if at least one of the materials is an insulator. Interestingly, this transfer of charge can even occur without frictional contact [37]. The earliest records of triboelectric phenomena date back to antiquity. Contrary to popular belief, the first detailed descriptions of electrostatic electricity were made by Plato in his *Timaeus*, not by Thales of Miletus, who only briefly mentioned it in one of his philosophical manuscripts [42]. Despite being known for such a long time, it is remarkable that our scientific understanding of this phenomenon has not significantly advanced. In this section we describe briefly several concepts about electrostatics and we emphasize their relevance in to our study subject which is the triboelectric phenomenon.

1.1.1 Principles of electrostatics

In electrostatics, a neutral object can acquire a positive or negative charge through the loss or gain of electrons or ions. Depending on the interaction between two charged particles, they will either attract if their charges are opposite or repel if their charges are the same. The strength of this attraction or repulsion, known as the electrostatic force, is quantified by Coulomb's law:

$$F_q = \frac{1}{4\pi\varepsilon_0\varepsilon_r}\frac{q_1 q_2}{r^2} \tag{1.1}$$

where F_q is the electrostatic force; ε_0 is the permittivity of a vacuum; ε_r is the relative permittivity of the medium (such as air); r is the distance between two point charges (e.g., two particles); and q_1 and q_2 are the charges of the two point sources. It is important to note that equation (1.1) considers only two charges. To calculate the electrostatic force exerted on a charge by multiple surrounding charges, one must sum the individual forces between the charge and each of the surrounding charges. Another key concept to consider is the electric field, defined as the electrostatic force per unit charge. The magnitude of the electric field, considering only a point charge q, is given by

$$E = \frac{q}{4\pi\varepsilon_0\varepsilon_r r^2} \tag{1.2}$$

1.1.1.1 Charge generation for insulator objects

In general, charging can occur either by contact or by induction. In contact charging, static charge accumulation happens during sliding, rolling, or impact between two objects [56]. In contrast, induction charging, also known as image charging, does not require direct contact. It occurs when a charged object is brought near an uncharged conductive object, causing polarization as electrons are redistributed within the uncharged object [84]. While induction charging has traditionally been associated with conductive materials, recent studies have shown that this effect can also occur in insulators if they remain in close proximity to a charged object for an extended period [1].

The challenge, however, lies in the fact that, unlike conductive materials, charging mechanisms in insulating materials have not been extensively studied due to their low charge mobility. This limitation confines triboelectric charging to a localized region on the surface of the object, typically within a microscopic range. The complexity of studying electrostatic effects further increases when dealing with non-spherical particles, which is often the case with polymers [29].

A commonly used method to demonstrate an object's tendency for triboelectric charging is known as the work function. This property essentially describes the amount of thermodynamic work required to remove an electron from the material's surface to a neutral point in the vacuum near the surface. Therefore, a material with a high work function has a greater likelihood of retaining electrons on its surface. Based on this property, we can define the contact potential difference, which is determined by the following formula:

$$V_C = \frac{(\phi_1 - \phi_2)}{e}, \tag{1.3}$$

where ϕ_1 and ϕ_2 are the work functions of the two metals/conductors, and e is the elementary charge. This contact potential difference arises after the electron flow at the contact interface has reached equilibrium (see Fig.1.1). On the other hand, the difference found in the numerator of expression (1.3) is known as the driving force for electron transfer from one surface to another.

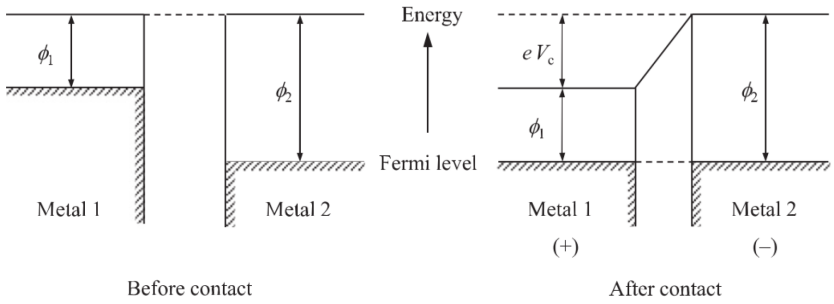


Figure 1.1: Electron potential energy for metal-metal contacts [56].

While the work function model is not initially considered applicable to metal-insulator and insulator-insulator contacts due to the lack of "free electrons" in insulators, experimental evidence has shown that electron transfer does occur in these types of contacts [22]. Consequently, the "effective" work function for polymers was determined through metal-polymer contacts [75].

1.1.1.2 Charge transfer modeling

The electrostatic charging tendencies of various materials have been thoroughly investigated through laboratory-scale insulator-metal contact experiments [27], [41], [57]. In addition to differences in effective work function, other factors influencing charge transfer include contact angle, initial charge, particle size, temperature, and relative humidity [56]. So far, the effects of these operational conditions have only been qualitatively analyzed, while material properties and collision characteristics have been incorporated into charge transfer models to estimate the rate of charge transfer per collision or contact duration.

A commonly used model for charge transfer is the condenser model, which assumes that the contact area between two surfaces, S_c, can be considered as a capacitor. This model uses the effective work function difference, V_c, as the fixed driving force to determine the magnitude and direction of charge transfer on a particle [55]:

$$\Delta q = C_e \frac{\epsilon_0 \epsilon_r V_c}{z_c} \left(1 - \frac{q_i}{q_\infty}\right) S_c \tag{1.4}$$

In this equation, C_e represents the charging efficiency, ϵ_0 is the absolute permittivity, ϵ_r is the relative permittivity of the medium, z_c is the critical gap, q_i is the particle's initial charge, and q_∞ is the charge saturation point or the maximum charge a particle can hold. This model, along with others, has been shown to reasonably describe electrostatic charge transfer in insulator-metal contacts, made possible by the identification of charge saturation points and effective work functions of the materials involved at a laboratory scale.

In contrast to this type of insulator-metal contact charging, the understanding of insulator-insulator collisions remains problematic, especially in the case of granular flows. Lacks and Shinbrot (2019) [47] found that our quantitative understanding of electrostatics is sometimes challenged by the complexity of granular materials. For instance, the same model cannot be directly applied to granular flows, whether involving the same or different materials, without modifications. Therefore, it is crucial to conduct experiments on particle-particle contact charging to better understand the electrostatic properties or material characteristics responsible for the complex charging behavior between particles. Given the empirical limitations, conducting numerical simulations is a promising approach due to its clear cost advantages.

1.2 Modeling and simulation of gas-solid fluidized beds

The rapid progression of computing technology has positioned computational fluid dynamics (CFD) modeling as a crucial tool for the systematic study and enhancement of gas-solid fluidized beds across various industries, including environmental, chemical, petrochemical, energy, and metallurgical sectors [69, 78]. The primary challenge in modeling industrial-scale fluidized beds lies in the significant scale separation between the large flow structures and the interactions between particles and the carrier gas, as well as particle collisions, which impact the overall flow dynamics. Since it is not feasible to resolve all spatial and temporal scales in a single simulation, gas-solid flow phenomena must be modeled using methods at different levels of detail [76]. These models are conveniently classified by separately treating the solid and gaseous phases. For each phase, two approaches are commonly used: Eulerian and Lagrangian. The Eulerian approach describes the phase as a continuum, typically using a Navier-Stokes equation. Conversely, the Lagrangian approach represents the phase as discrete volume governed by Newton's second law. By combining these approaches for each phase, Table 1.1 presents a classification of the most commonly used models in gas-solid flows. Figure 1.2 complements this by providing a visual illustration of each model.

Name	**Gas phase**	**Solid phase**	**Scale**
1. Two-fluid model	Eulerian	Eulerian	Engineering (1 m)
2. Discrete element model (CFD-DEM)	Eulerian (unresolved)	Lagrangian	Laboratory (0.1 m)
3. Direct numerical simulation (PR-DNS)	Eulerian (resolved)	Lagrangian	Laboratory (0.01 m)

Table 1.1: Classification of the various models used for simulating dense gas-solid flow in the context of gas-solid flows [76].

The Two-Fluid Model (TFM) is an Euler-Euler framework that treats both phases as continuous media, making it ideal for describing two-phase flows at relatively large scales. In this model, the continuity, momentum, and energy equations are solved independently for each phase, incorporating models for interfacial transfer of certain physical quantities, such as mass, momentum, energy, temperature, etc.,

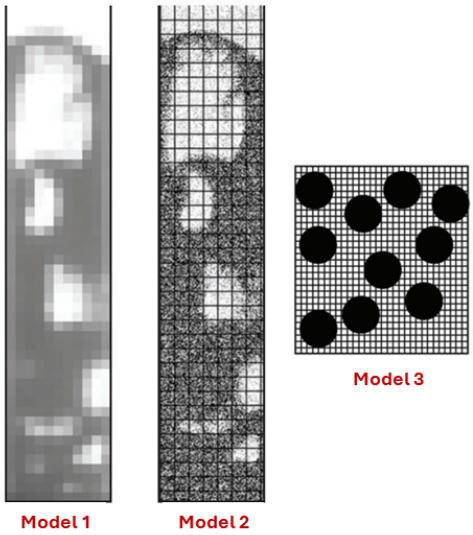


Figure 1.2: Graphical representation of the models listed in Table 1.1. Only models 1 and 2 (the two-fluid model and the discrete element model) are employed for the simulation of actual gas-fluidized beds. Model 3 (direct numerical simulation model) is used for small, representative parts of fluidized beds [76].

between the two phases. The fluid dynamics of the granular phase are determined using closures derived from the kinetic theory of granular flow, which is still an active area of research and development [76].

Despite its computational efficiency, a significant limitation of TFM is its inability to accurately capture detailed particle-fluid interactions, as well as the complexities of particle-particle and particle-wall collisions. To address these challenges, smaller-scale modeling can greatly contribute to the development and enhancement of continuous, average-based macroscopic models, as these are often employed as research tools to gain deeper insights into particle-gas interactions within the flow [51].

In discrete element models (CFD-DEM), the Eulerian grid cells are typically at least 10 times larger than the particle diameter, which simplifies gas-particle interactions by approximating particles as point sources for mass and heat transfer and as localized regions where momentum is removed from the fluid phase. This approximation requires the use of empirical correlations for gas-solid interactions, similar to those used in TFM [76]. In contrast, particle-resolved direct numerical simulations (PR-DNS) use a grid that is at least 10 times smaller than the particle size, enabling the direct resolution of solid-gas interactions. This approach allows for high-fidelity microscale simulations, where complex interactions in gas-solid flows can be accurately captured and studied in detail.

In the context of electrostatic modeling in gas-solid flows, such as fluidized beds in the polyolefin industry, Euler-Euler simulations are essential for optimizing or designing reactors. These simulations are based on large-scale models that must incorporate a charge transfer model for insulating particles (such as polypropylene used in the polyolefin industry). Such large-scale models are developed using CFD-DEM simulations, which, as we have seen, operate at laboratory scale. At this scale, it is possible to track pointwise the charge of the particles. There are several studies have investigated the triboelectric modeling of particle-laden flows at this scale [80], [52], [73]. Among the most notable works on the electrification of insulating particles is that of Sippola et al. [73], who developed a CFD-DEM model considering the effective work function difference between materials and the electric field at the contact point. However, the model showed limitations as the simulation results did not precisely match the experimental data for all input parameters. This discrepancy could be due to the lack of consideration of particle polarity, which has been shown to affect particle interactions [50].

To develop mesoscale models, it is necessary to use sub-models that account for such polarity as well as other electrostatic quantities such as effective work function and electric field. These models are obtained using PR-DNS simulations at the microscopic scale, where not only the particle charge but also the charge distribution over the particle surface can be tracked. This is particularly important for insulating particles, as the surface charge distribution plays a critical role, as discussed earlier.

Chapter 2

Viscous penalty method with fully coupled solver

The purpose of this chapter is to provide a concise overview of a numerical modeling strategy for Direct Numerical Simulation (DNS) of dense particle flows, known as the viscous penalty method, utilizing a fully coupled solver. Additionally, it will briefly address the treatment of particle advection and solid collisions, which are essential for simulating resolved-scale particle flows—i.e., scenarios where the particles and the fluid share similar characteristic scales. The primary references for this section include [11], [16], [62], and [79].

2.1 Viscous penalty method

In the numerical simulation of particle-laden flows, two primary strategies are commonly adopted. The first strategy involves using unstructured body-fitted in order to cover the fluid domain only. This requires that grids rely on automatic mesh generation as solid particles move through time and space [79]. This approach, while intuitive, faces significant challenges: it is computationally intensive and difficult to implement in software, especially for complex fluid-solid interfaces [24]. Examples of this method include the Arbitrary Lagrangian-Eulerian (ALE) framework [40, 49, 58]. Alternatively, there is a strategy known as the structured grid approach, which involves maintaining a fixed mesh that includes both phases without adapting to their interfaces. However, this approach presents a challenge since the solid-fluid interface cannot be explicitly tracked by the non-conforming mesh, leading to computational errors near the moving interface [53]. This approach serves as the basis for the Viscous Penalty Method, which will be explained later, and it is part of a broader category of modeling techniques known as fictitious domain methods [34, 46].

In our study, we employ a one-fluid model coupled with the Volume of Fluid (VOF) method, which uses a phase function, C, to distinguish between solid and fluid regions within the flow. The values of C are used to determine the specific rheological properties (such as density or viscosity) required to differentiate the solid phase from the fluid phase (see Figure 2.1). By definition, $C = 1$ corresponds to the solid phase, $C = 0$ to the fluid phase, and $C = 0.5$ to the interface between them.

For modeling incompressible two-phase flows involving a carrier fluid and a solid phase, the following incompressible Navier-Stokes equations are solved:

$$\nabla \cdot \mathbf{u} = 0, \tag{2.1}$$

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) = -\nabla p + \rho \mathbf{g} + \nabla \cdot \left(\mu \left(\nabla \mathbf{u} + \nabla^t \mathbf{u}\right)\right) + \mathbf{F}_{si}, \tag{2.2}$$

where $\mathbf{u}$ represents the velocity, p the pressure, t the time, and $\mathbf{g}$ the gravity vector. The parameters ρ and μ denote the density and viscosity of the equivalent fluid, respectively, both of which depend on the value of C. The term $\mathbf{F}_{si}$ represents a volumetric force that accounts for solid interactions, activated during collisions between particles or between particles and walls, ensuring four-way coupling between the phases. For particle advection, the VOF method first updates particle positions in a Lagrangian framework and then computes the Eulerian solid fractions in the mesh by projecting the particle shapes onto the fluid grid (see section § 2.1.3).

Other approaches, such as the ALE framework, involve solving the Navier-Stokes equations for the fluid phase and the Newton-Euler equations for the particles, then connecting these solutions at the solid-fluid interface using jump equations (see, for example, [58, 23]). In contrast, our approach implicitly

incorporates the jump relations within the one-fluid model (i.e., equations (2.1) and (2.2)), as described in [43].

Let us remark the similarity between the classical incompressible Navier-Stokes equations and the one-fluid model. The differences lie in the fact that the latter introduces a volumetric force to account for collision effects, and both the viscosity μ and density ρ depend on the phase function C. Regarding the physical constraints that must be applied to the solids, a specific model based on viscosity penalization is required, which will be presented in the following section.

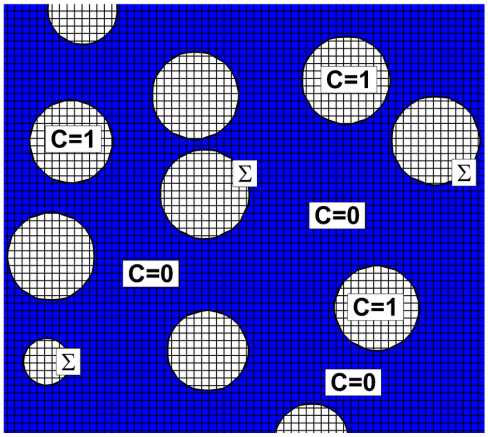


Figure 2.1: Sketch of fictitious domain approach for a finite size particulate flow. The solid-fluid interface Σ, the phase function C and the underlying grid are showed [79].

2.1.1 Penalty methods for solid behavior and incompressibility

Once our one-fluid model and the phase function C have been defined, it is natural to inquire about the specific rheological law needed to ensure, for instance, solid-like behavior (i.e., when $C = 1$). Additionally, it should be noted that penalizing the velocity is not feasible in this context, as we are dealing with problems such as fluidized beds or particle sedimentation, where particle velocities are not known a priori.

Following the work of Vincent et al. (2014) [79], the approach employed to ensure solid particle behavior involves decomposing the stress tensor σ of a Newtonian fluid (see [36] and [67]) as follows:

$$\sigma_{ij} = -p\delta_{ij} + \lambda(\nabla \cdot \mathbf{u})\delta_{ij} + 2\mu\mathbf{D}_{ij}$$

where λ and μ represent the compressional and shear viscosities, respectively, and $\mathbf{D}$ is the rate-of-deformation tensor.

To further distinguish the different contributions of viscosity—compression λ, tearing κ, shearing ζ, and rotation η—we can reformulate the stress tensor as proposed by Caltagirone (2001) [15], as shown in the following expression:

$$\begin{aligned}\sigma = &\begin{bmatrix} -p+\lambda\nabla\cdot\mathbf{u} & 0 & 0 \\ 0 & -p+\lambda\nabla\cdot\mathbf{u} & 0 \\ 0 & 0 & -p+\lambda\nabla\cdot\mathbf{u} \end{bmatrix} + \kappa \begin{bmatrix} \frac{\partial u}{\partial x} & 0 & 0 \\ 0 & \frac{\partial v}{\partial y} & 0 \\ 0 & 0 & \frac{\partial w}{\partial z} \end{bmatrix} \\ &+ \zeta \begin{bmatrix} 0 & \frac{\partial u}{\partial y} & \frac{\partial u}{\partial z} \\ \frac{\partial v}{\partial x} & 0 & \frac{\partial v}{\partial z} \\ \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} & 0 \end{bmatrix} - \eta \begin{bmatrix} 0 & \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} & \frac{\partial u}{\partial z} - \frac{\partial w}{\partial x} \\ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} & 0 & \frac{\partial v}{\partial z} - \frac{\partial w}{\partial y} \\ \frac{\partial w}{\partial x} - \frac{\partial u}{\partial z} & \frac{\partial w}{\partial y} - \frac{\partial v}{\partial z} & 0 \end{bmatrix}\end{aligned} \tag{2.3}$$

which can be written in a compact form as

$$\sigma_{ij} = (-p + \lambda\nabla \cdot \mathbf{u})\delta_{ij} + \kappa\Lambda_{ij} + \zeta\Theta_{ij} - \eta\Gamma_{ij}. \tag{2.4}$$

The advantage of this formulation is that by modifying the viscosity coefficients associated with each term directly within the Navier–Stokes equations for a Newtonian fluid, it becomes possible to independently control the magnitude of each contribution of viscosity. This allows for more flexible and

precise modeling of fluid behavior. Since we are considering that the two fluids are incompressible, we set $\lambda = 0$, and we note that the viscous tress tensor in (2.2) can be expressed as follows

$$\nabla \cdot \left(\mu \left(\nabla \mathbf{u} + \nabla^t \mathbf{u}\right)\right) = \nabla \cdot [\kappa \mathbf{\Lambda}(\mathbf{u})] + \nabla \cdot [\zeta \mathbf{\Theta}(\mathbf{u})] - \nabla \cdot [\eta \mathbf{\Gamma}(\mathbf{u})]. \tag{2.5}$$

To impose a solid behavior, we set no tearing, no shearing, and a rotational viscosity consistent with the surrounding fluid, i.e., $\kappa = 2\eta$, $\zeta = 2\eta$, and $\eta \gg 1$ in regions where $C = 1$. Although this configuration is mathematically equivalent when $\mu \gg 1$, differences emerge from a discrete perspective. On a staggered grid, using a single viscosity contribution leads to a rasterization effect (see Fig. 2.2) due to interpolation errors when computing shearing and rotational viscosities. Indeed, The approach of using distinct viscosity contributions achieves second-order convergence, as demonstrated by Vincent et al. (2014) [79].

In the following section, we will justify the practical consideration of using very high viscosity values, μ. This is because the property of free deformation proves to be both sufficient and necessary for the equivalent fluid behavior of the solid.

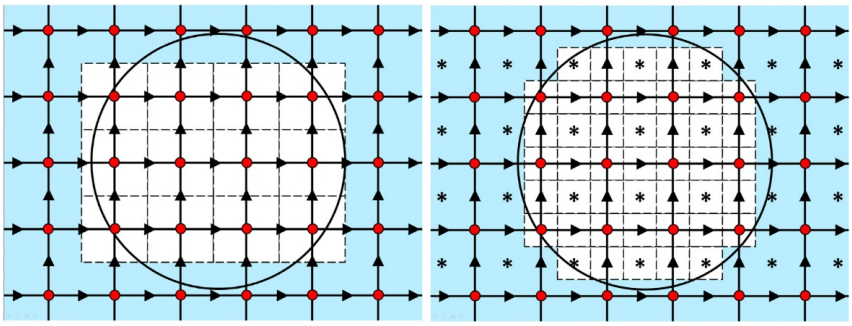


Figure 2.2: A discrete representation of the direct viscous penalty method (left) and the split viscous penalty method (right) on staggered grids – pressure points are depicted with red circles, velocities with arrows, and pure shearing and rotational viscosities with stars. The black line indicates the interface between a particle and the carrier fluid. [79].

2.1.2 Physical characteristics of the equivalent fluid and numeric implementation

The behavior of the solid phase is particularly characterized by specific rheological properties that govern its deformation and interaction with the surrounding fluid. Understanding these properties and their mathematical representation is crucial for implementing solid constraints and accurately capturing the dynamics of the fluid-solid interface. In this section, we will justify the rheological properties related to free deformation in the solid equivalent fluid as defined in our one-fluid formulation. Consider any point $\mathbf{P}$ within the solid equivalent fluid of our domain $\mathbf{\Omega}$. For any particle i, whose related solid subdomain is denoted as $\mathbf{\Omega^i}$, the velocity $\mathbf{u}$ at point $\mathbf{P}$ can be described by translation $\mathbf{u}_0$ and rotation ω components as follows:

$$\forall \mathbf{P} \in \mathbf{\Omega}^i, \exists \{\mathbf{w}, \mathbf{u}_0\} \in \mathbb{R}^d \times \mathbb{R}^d, \quad \mathbf{u}(\mathbf{P}) = \mathbf{u}_0 + \mathbf{X^i_{BC}} \mathbf{P} \wedge \omega \tag{2.6}$$

where $\mathbf{X^i_{BC}}$ is the barycenter of particle i.

Now we aim to proof that the expression (2.6) is equivalent to the deformation tensor is zero, i.e.

$$\forall \mathbf{P} \in \mathbf{\Omega}^i, \quad \nabla \mathbf{u} + \nabla^T \mathbf{u} = 0 \tag{2.7}$$

In fact, it is clear that (2.6) implies (2.7). Conversely, let us suppose that the above expression holds. Then we obtain the following systems:

$$\begin{cases} \dfrac{\partial u}{\partial x} = 0 \\ \dfrac{\partial v}{\partial y} = 0 \\ \dfrac{\partial w}{\partial z} = 0 \end{cases} \tag{2.8}$$

$$\begin{cases} \dfrac{\partial u}{\partial y} = -\dfrac{\partial v}{\partial x} \\ \dfrac{\partial v}{\partial z} = -\dfrac{\partial w}{\partial y} \\ \dfrac{\partial w}{\partial x} = -\dfrac{\partial u}{\partial z} \end{cases} \tag{2.9}$$

Let us notice that the system (2.8) provide us the different dependecies of the velocity's component. With this in mind and the second system (2.9) we obtain the resulting system:

$$\begin{cases} \dfrac{\partial u(y,z)}{\partial y} = -\dfrac{\partial v(x,z)}{\partial x} = \omega_3(z) \\ \dfrac{\partial v(x,z)}{\partial z} = -\dfrac{\partial w(x,y)}{\partial y} = \omega_1(x) \\ \dfrac{\partial w(x,y)}{\partial x} = -\dfrac{\partial u(y,z)}{\partial z} = \omega_2(y) \end{cases}$$

If we assume that the rotation velocity components have sufficient regularity, by the Schwarz theorem, we find that the second derivatives of the velocity components are constant, i.e. there exists $W_1, W_2, W_3 \in \mathbb{R}$ such that

$$\begin{cases} \dfrac{\partial^2 u(y,z)}{\partial y \partial z} = \dfrac{\partial \omega_3(z)}{\partial z} = -\dfrac{\partial \omega_2(y)}{\partial y} = W_1 \\ \dfrac{\partial^2 v(x,z)}{\partial x \partial z} = \dfrac{\partial \omega_1(x)}{\partial x} = -\dfrac{\partial \omega_3(z)}{\partial y} = W_2 \\ \dfrac{\partial^2 w(x,y)}{\partial x \partial y} = \dfrac{\partial \omega_2(y)}{\partial y} = -\dfrac{\partial \omega_1(x)}{\partial x} = W_3. \end{cases} \tag{2.10}$$

This implies that the fonctions ω_1, ω_2 and ω_3 are constant, and finally, by integrating the system's equations in (2.10) we get:

$$\begin{cases} u = u_0 + \omega_3 y - \omega_2 z \\ v = v_0 + \omega_1 y - \omega_3 z \\ w = w_0 + \omega_2 y - \omega_1 z. \end{cases}$$

Therefore, we conclude that the relations (2.6) and (2.7) are equivalent. This equivalence implies that the solid constraint can be established using $\nabla \mathbf{u} + \nabla^T \mathbf{u} = 0$, and this condition is asymptotically satisfied as $\mu \to +\infty$. Moreover, it can be verified that $\nabla \mathbf{u} + \nabla^T \mathbf{u} = 2\Lambda(\mathbf{u}) + 2\Theta(\mathbf{u}) - \Gamma(\mathbf{u})$. Thus, another equivalent relation of (2.6) and (2.7) is given by:

$$\Lambda(\mathbf{u}) = 0 \quad \text{and} \quad \Gamma(\mathbf{u}) = 2\Theta(\mathbf{u}). \tag{2.11}$$

For the numerical implementation, three types of mesh cells are considered, depending on the regions they occupy within the particle-laden flow. The first and second types consist of mesh cells that are fully contained within either the solid medium (first type) or the Newtonian fluid (second type). In both cases, it is crucial that the conditions $\kappa = 2\mu$, $\zeta = 2\mu$, and $\eta = \mu$ are satisfied. For the first type (within the solid phase), it is particularly important that $\mu \gg 1$. In practice, for this case, the viscosity μ is typically chosen to be 1000 to 10,000 times greater than that of the carrier fluid, depending on the computational cost of the linear solvers (see [79]).

The third type of grid cell is characterized by the fluid-solid interface Σ passing through it. For this scenario, various methods exist to determine a homogenized viscosity, collectively referred to as numerical viscous laws. These laws depend on the viscosities of the solid (μ_s) and the fluid (μ_f), as well as the values of the phase functions for both the diagonal (C) and off-diagonal (C_μ) terms of the viscous stress tensor in (2.3). Here, we present three of the most commonly used viscous laws in the literature:

1) Discontinuous law:

$$\begin{aligned} \kappa &= 2\left[\mu_f I_{C<0.5} + \mu_S I_{C\geqslant 0.5}\right] \\ \zeta &= 2\left[\mu_f I_{C_\mu<0.5} + \mu_S I_{C_\mu\geqslant 0.5}\right] \\ \eta &= \mu_f I_{C_\mu<0.5} + \mu_S I_{C_\mu\geqslant 0.5} \end{aligned}$$

2) Arithmetic law:

$$\begin{aligned} \kappa &= 2\left[(1-C)\mu_f + C\mu_s\right] \\ \zeta &= 2\left[(1-C_\mu)\,\mu_f + C_\mu\mu_s\right] \\ \eta &= (1-C_\mu)\,\mu_f + C_\mu\mu_s \end{aligned}$$

3) Harmonic law:

$$\kappa = 2\left[\frac{\mu_f \mu_s}{C\mu_f + (1-C)\mu_s}\right]$$
$$\zeta = 2\left[\frac{\mu_f \mu_s}{C_\mu \mu_f + (1-C_\mu)\,\mu_s}\right]$$
$$\eta = \frac{\mu_f \mu_s}{C_\mu \mu_f + (1-C_\mu)\,\mu_s}$$

where I_C is a condition indicator function satisfying $I_{C<0.5}$ if $C < 0.5$ or $I_{C>0.5} = 1$ if $C \geq 0.5$. Now following the results of Chadil et al. (2019) [16], we will use a viscous law formed by an harmonic average where C_μ and C is defined by projecting the particle shape with the virtual point procedure (see section § 2.1.3.2 for more details). Moreover, for the density computation we will use an arithmetic average over the whole discretisation grid. This formula is given by

$$\rho = (1-C)\rho_f + C\rho_s$$

where ρ_s and ρ_f are the density values at the solid and fluid medium respectively.

Once established rheological laws that distinguish the solid particles from the carrier fluid, we will now proceed to explain the process by which particles are advected within our fluid grid in the following section.

2.1.3 Eulerian–Lagrangian VOF method for particle tracking

The accurate modeling of particle transport and distribution within a fluid flow is a critical aspect of simulating multiphase flows. In this section, we will describe the process by which finite-sized particles are transported and how their solid fraction C within the Eulerian flow. Within the general framework, this step occurs after solving the mass and momentum equations (including particle collision $\mathbf{F}_{si}$) using a method that ensures coupling between velocity and pressure. At this stage, instead of solving the advection equation to determine the new solid fraction of the particle in the Eulerian cells using the classical Eulerian VOF approach, the process involves using the Lagrangian velocity of the particle to update its position. The solid fraction is then calculated by projecting the exact shape of the particle onto the flow grid. This approach is employed to avoid the shape distortions that would otherwise be introduced by this conventional method.

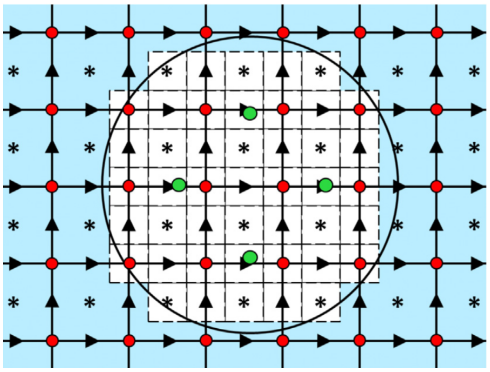


Figure 2.3: Position in two dimensions of the reference Lagrangian points (green points) used to interpolate the Lagrangian velocity at the barycenter of the particles [79].

2.1.3.1 Computation of Lagrangian particle's velocity

The process of transporting finite-sized particles begins with calculating the Lagrangian velocity, assuming that the particle's center of mass is already known from prior computations. To achieve a more accurate velocity calculation, reference points within each sphere are used instead of averaging the Eulerian velocity field over the solid sphere, with four points in two dimensions and six in three dimensions (see [79]). These points are positioned parallel to the velocity axis at a distance of half the particle's radius from the sphere's center, as shown in Fig. 2.3. The Lagrangian velocity $\mathbf{V}^{I,L}$ of the particle is then calculated by interpolating the velocities at these reference points, whose positions are denoted as $\mathbf{X_k}^{I,L}$. Here, I represents the particle index, and k corresponds to the index of the N interpolation points, where $N = 4$ in 2D or $N = 6$ in 3D.

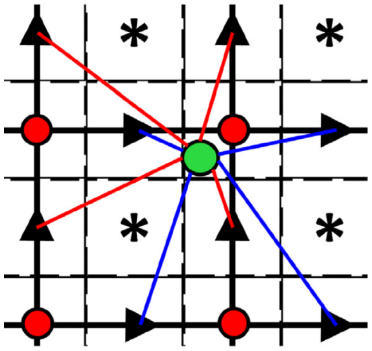


Figure 2.4: Eulerian velocity kernel (u-components in blue and v-components in red) used in two dimensions to estimate the components of the interpolation Lagrangian velocity points (green point)[79].

The different velocities for interpolating the Lagrangian velocity are obtained through interpolation using the nearest Eulerian velocity values, as shown in two dimensions in Fig. 2.4. The first step is to calculate the relative positions $\tilde{\mathbf{X}}$ of the interpolation points with respect to the Eulerian velocity components $\mathbf{X_u}$ and the reference points $\mathbf{X}_k^{I,L}$ as follows

$$\tilde{\mathbf{X}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{pmatrix} = \frac{\mathbf{X}_k^{I,L} - \mathbf{X_u}}{\mathbf{DX}^{I,L}},$$

where for a given interpolated Lagrangian velocity component, $\mathbf{DX}^{I,L}$ represents the coordinate-wise distance from the sides of the box formed by the four closest Eulerian velocity positions in two dimensions (or six positions in three dimensions) for any velocity component. In two dimensions, this could refer to the horizontal velocity component u or the vertical velocity component v, while in three dimensions, an additional velocity component exists. This concept is illustrated in two dimensions in Fig. 2.4, where the four closest points of the two components of the Eulerian velocity (the horizontal component u and the vertical component v) are connected to a reference point (green node) by blue lines for the u component, and red lines for the v component.

To determine each Eulerian velocity component at the reference points, we aim to compute the basis functions P_m, $m = 1, \ldots, N$, for the bilinear interpolation:

$$\begin{aligned} V_{k,l}^{I,L} &= P_1 + P_2\tilde{x} + P_3\tilde{y} + P_4\tilde{x}\tilde{y} \quad \text{in 2D}, \\ V_{k,l}^{I,L} &= P_1 + P_2\tilde{x} + P_3\tilde{y} + P_4\tilde{z} + P_5\tilde{x}\tilde{y} + P_6\tilde{x}\tilde{z} + P_7\tilde{z}\tilde{y} + P_8\tilde{x}\tilde{y}\tilde{z} \quad \text{in 3D}, \end{aligned}$$

for the l^{th} Lagrangian velocity component, we consider the $n = 4$ nearest Eulerian velocities for the two components (u and v) in two dimensions, and the $n = 6$ nearest Eulerian velocities for the three components (u, v, and w) in three dimensions. For both cases, we denote these velocities as V_n. Thus, we obtain:

$$\begin{aligned} P_1 &= V_1, \\ P_2 &= V_2 - V_1, \\ P_3 &= V_4 - V_1, \\ P_4 &= V_1 + V_3 - V_2 - V_4. \end{aligned}$$

in two dimensions and

$$\begin{aligned} P_1 &= V_1 \\ P_2 &= V_2 - V_1 \\ P_3 &= V_4 - V_1 \\ P_4 &= V_5 - V_1 \\ P_5 &= V_3 + V_1 - V_2 - V_4 \\ P_6 &= V_6 + V_1 - V_2 - V_5 \\ P_7 &= V_8 + V_1 - V_4 - V_5 \\ P_8 &= V_7 - V_6 - V_8 + V_5 \end{aligned}$$

in three dimensions. Then we finally obtain the Lagrangian velocity of the particle as follows:

$$\mathbf{V}^{I,L} = \frac{\sum_N \mathbf{V}_k^{I,L}}{N}.$$

Finally, the next step is to update the position of the particle $\mathbf{X^{I,L}}$ by using our Lagrangian velocity as follows

$$\mathbf{X}^{I,L,n+1} = \mathbf{X}^{I,L,n} + \Delta t \mathbf{V}^{I,L,n+1/2} \tag{2.12}$$

with $\mathbf{V}^{I,L,n+1}$ as a second order Lagrangian velocity extrapolation at time $(n+1)\Delta t$ given by

$$\mathbf{V}^{I,L,n+1/2} = \frac{\mathbf{V}^{I,L,n} + \mathbf{V}^{I,L,n+1}}{2}, \text{ where } \mathbf{V}^{I,L,n+1} = \frac{\sum_N \mathbf{V}_k^{I,L,n+1}}{N}.$$

Here, $\mathbf{V}_k^{I,L,n+1}$ is defined by extrapolating the position of the particle at time $(n+1)\Delta t$, given by $\mathbf{X}^{I,L,n+1} = \mathbf{X}^{I,L,n} + \Delta t \mathbf{V}^{I,L,n+1}$.

2.1.3.2 Update of the solid fraction

Once the new position of the particle is determined by calculating the Lagrangian velocity, it is time to project the shape of the particle at the new position. This is done through a process called virtual testing, which involves creating a regular grid of test points within the control volume for pressure (where the function C is defined) and calculating the solid fraction, given that the radius of the solid sphere is known. For example, in Fig. 2.5, we observe that 100 virtual points are used to determine the solid fraction C^{n+1} (we consider the particle's position at time $(n+1)\Delta t$). The green points represent those inside the particle, while the blue points represent those outside. In this case, the solid fraction is $C^{n+1} = 92/100 = 0.92$.

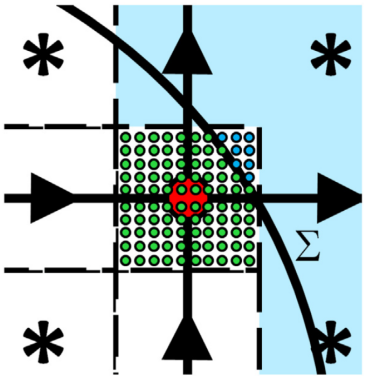


Figure 2.5: Virtual test method[79].

Additionally, the study by Vincent et al. (2014) [79] has shown that employing 25^{Ndim} virtual points (where Ndim represents the number of dimensions) ensures that the error in calculating the solid fraction consistently stays below 0.1%, regardless of the flow grid. This precision is essential for direct numerical simulations (DNS) of particulate flows, as it permits the use of a limited number of Eulerian cells across the particle diameter in three-dimensional simulations, making the modeling of realistic scenarios feasible.

2.2 Fully coupled solver

In this section, we provide a concise overview of the method used to generate the matrix system derived from equations (2.1) and (2.2) for the two-dimensional case, along with a brief presentation of the methodology to solve it. This approach is based on the work of Mohamed El Ouafa (2022) [62], which also covers the three-dimensional case and provides a detailed explanation of the solver used and matrix preconditioning. Following this work, an important consideration regarding the boundary conditions is that we will use a slightly modified version of our initial system:

$$\frac{\partial \rho \mathbf{u}}{\partial t} + \nabla \cdot (\rho \mathbf{u} \otimes \mathbf{u}) + \overline{\overline{B}}_{penu} (f(\mathbf{u}) - \mathbf{u}_\infty) = -\nabla p + \nabla \cdot \left(\mu \left(\nabla \mathbf{u} + \nabla^t \mathbf{u}\right)\right) + \rho \mathbf{g} + \mathbf{F}_{si} \tag{2.13}$$

$$\nabla \cdot \mathbf{u} = 0, \tag{2.14}$$

where

$$\overline{\overline{B}}_{\text{penu}} = \begin{pmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{pmatrix}.$$

In this equation, the term $\overline{\overline{B}}_{penu}\left(f(\mathbf{u})-\mathbf{u}_\infty\right)$ allows to define the boundary conditions [2]. This term will be detailed later. To begin our time-space discretization, let us first expand the initial two terms of (2.13) and apply the conservation of mass:

$$\begin{aligned}\frac{\partial(\rho\mathbf{u})}{\partial t}+\nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u}) &= \rho\frac{\partial\mathbf{u}}{\partial t}+\nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u})+\mathbf{u}\frac{\partial\rho}{\partial t}+\mathbf{u}\nabla\cdot(\rho\mathbf{u})-\mathbf{u}\nabla\cdot(\rho\mathbf{u})\\ &= \rho\frac{\partial\mathbf{u}}{\partial t}+\nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u})-\mathbf{u}\nabla\cdot(\rho\mathbf{u})+\mathbf{u}\left(\frac{\partial\rho}{\partial t}+\nabla\cdot(\rho\mathbf{u})\right)\end{aligned}$$

We will therefore work with the following equations:

$$\rho\frac{\partial\mathbf{u}}{\partial t}+\nabla\cdot(\rho\mathbf{u}\otimes\mathbf{u})-\mathbf{u}\nabla\cdot(\rho\mathbf{u})+\overline{\overline{B}}_{\text{penu}}\left(f(\mathbf{u})-\mathbf{u}_\infty\right)=-\nabla p+\nabla\cdot\left(\mu\left(\nabla\mathbf{u}+\nabla^t\mathbf{u}\right)\right)+\rho\mathbf{g}+\mathbf{F}_{si} \quad (2.15)$$

$$\nabla\cdot\mathbf{u}=0. \quad (2.16)$$

2.2.1 Time integration of two-phase Navier-Stokes equations

Since we are treating with time dependent equations, the time step Δt plays an important role. Here, the time integration is added by considering Euler schemes of order 1 or 2. Hence, we obtain the following semi-discret system:

$$\begin{aligned}&\frac{\rho^{n+1}}{\Delta t}\left(\alpha\mathbf{u}^{n+1}-\beta\mathbf{u}^n+\gamma\mathbf{u}^{n-1}\right)+\nabla\cdot(\rho^{n+1}\mathbf{u}^{n+1}\otimes\mathbf{u}^{n+1})-\mathbf{u}^{n+1}\nabla\cdot(\rho^{n+1}\mathbf{u}^{n+1})\\ &\quad+\overline{\overline{B}}_{penu}\cdot\left(f\left(\mathbf{u}^{n+1}\right)-\mathbf{u}_\infty\right)=-\nabla p^{n+1}+\nabla\cdot\left(\mu\left(\nabla\mathbf{u}^{n+1}+\nabla^t\mathbf{u}^{n+1}\right)\right)+\rho^{n+1}\bar{g}+\mathbf{F}_{si}, \quad (2.17)\end{aligned}$$

$$\nabla\cdot\mathbf{u}^{n+1}=0, \quad (2.18)$$

where the parameters α,β et γ are given by the following manner :

- $\alpha=1,\beta=-1$ et $\gamma=0$: for the Euler's scheme of first order.
- $\alpha=3/2,\beta=-2$ et $\gamma=1/2$: for the Adam-bashfort extrapolation.

2.2.2 Spatial integration of two-phase Navier-Stokes equations

After handling the temporal integration, we move on to spatial discretization. For this, we use a staggered grid where velocity and pressure variables are represented on staggered points (see Fig. 2.6), each with its own control volume. The main advantage of this grid type is its ability to conserve fluxes, making the method conservative. Additionally, staggered grids prevent oscillations in the pressure field and allow for a natural discretization of differential operators. Boundary conditions are managed using ghost nodes outside the physical domain.

In this section, we present the spatial discretization in two dimensions; however, the procedure can be generalized to three dimensions. A key consideration in our approach is the use of a first-order Euler scheme for temporal discretization. Additionally, the finite volume method is employed to ensure the conservation of mass and momentum within each control volume [33]. This method involves integrating the conservation equations (see Eqs. (2.17) and (2.18)) over each control volume. We will now focus on the control volume for the u velocity component, $\Omega_{u_{(i,j)}}$, followed by the pressure, $\Omega_{p_{(i,j)}}$. A similar approach is applied to the v velocity component, yielding the following expressions:

$$\begin{aligned}&\int_{\Omega_{u_{(i,j)}}}\frac{\rho^n}{\Delta t}\left(u^{n+1}-u^n\right)dV+\int_{\Sigma_{u_{(i,j)}}}\nabla\cdot\left(\rho^n u^{n+1}\begin{pmatrix}u^n\\v^n\end{pmatrix}\right)dS-\int_{\Omega_{u_{(i,j)}}}u^{n+1}\nabla\cdot(\rho^n\mathbf{u}^n)\,dV\\ &\quad+\int_{\Omega_{u_{(i,j)}}}\begin{pmatrix}1\\0\end{pmatrix}^t\cdot\overline{\overline{B}}_{\text{penu}}\left(f\left(\mathbf{u}^{n+1}\right)-u_\infty\right)dV=-\int_{\Omega_{u_{(i,j)}}}\partial_x p^{n+1}dV \quad (2.19)\\ &\quad+\int_{\Sigma_{u_{(i,j)}}}\begin{pmatrix}1\\0\end{pmatrix}^t\cdot\left(\nabla\cdot\left(\mu\left(\nabla u^{n+1}+\nabla^t u^{n+1}\right)\right)\cdot\bar{n}\right)dS+\int_{\Omega_{u_{(i,j)}}}\begin{pmatrix}1\\0\end{pmatrix}^t\cdot\left(\rho^n\bar{g}dV+\mathbf{F}_{si}\right)dV\end{aligned}$$

$$\int_{\Omega_p}\nabla\cdot\mathbf{u}^{n+1}dV=0 \quad (2.20)$$

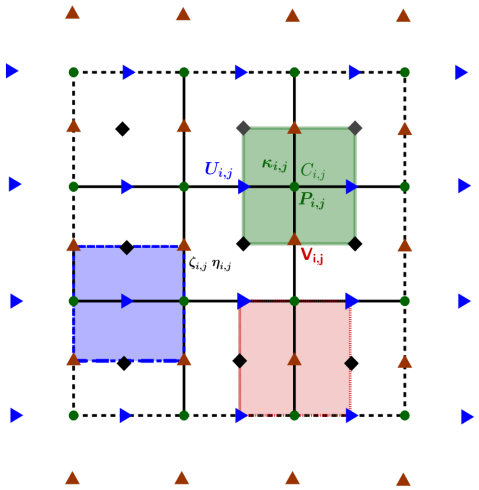


Figure 2.6: Staggered grid type. Control volumes utilized. The pressure $p_{i,j}$, elongation viscosity $\kappa_{i,j}$, and color function $C_{i,j}$ are located at the nodes of the Eulerian grid. The horizontal velocity $u_{i,j}$ is defined on the left and right faces of the control volume (in green) associated with $p_{i,j}$, while the vertical velocity $v_{i,j}$ is defined on the lower and upper sides of the control volume associated with $p_{i,j}$. Vorticity $\Omega_{i,j}$ and the shear and rotation viscosities $\eta_{i,j}$ and $\zeta_{i,j}$ are positioned at the corners of the control volume associated with $p_{i,j}$.

By using the divergence theorem we transform some volume terms from the previous expression into surface terms as follows:

$$\int_{\Omega_{u_{(i,j)}}} \frac{\rho^n}{\Delta t}\left(u^{n+1}-u^n\right)dV + \int_{\Sigma_{u_{(i,j)}}} \left(\rho^n u^{n+1}\begin{pmatrix}u^n\\ v^n\end{pmatrix}\right)\cdot \bar{n}\, dS - \int_{\Omega_{u_{(i,j)}}} u^{n+1}\nabla\cdot\left(\rho^n \mathbf{u}^n\right)dV$$
$$+ \int_{\Omega_{u_{(i,j)}}} \begin{pmatrix}1\\0\end{pmatrix}^t \cdot \overline{\overline{B}}_{\text{penu}}\left(f\left(\mathbf{u}^{n+1}\right)-u_\infty\right)dV = -\int_{\Omega_{u_{(i,j)}}} \partial_x p^{n+1} dV \tag{2.21}$$
$$+ \int_{\Omega_{u_{(i,j)}}} \begin{pmatrix}1\\0\end{pmatrix}^t \cdot \left(\rho^n \bar{g} dV + \mathbf{F}_{si}\right)dV + \int_{\Sigma_{u_{(i,j)}}} \begin{pmatrix}1\\0\end{pmatrix}^t \cdot \left(\left(\mu\left(\nabla u^{n+1}+\nabla^t u^{n+1}\right)\right)\cdot \bar{n}\right)dS$$

$$\int_{\Omega_p} \nabla\cdot \mathbf{u}^{n+1} dV = 0 \tag{2.22}$$

Now in the following subsections, we are going to discretize each integral term separately.

Convective terms

According to Fig. 2.7, we calculate the surface integral of the convective terms. For the first term, we obtain the following four contributions corresponding to the four sides of the control volume:

$$\int_{\Sigma_{u_{(i,j)}}} \left(\rho^n u^{n+1}\begin{pmatrix}u^n\\ v^n\end{pmatrix}\right)\cdot \bar{n}\, dS = \int_{\text{right}} \rho^n u^{n+1} u^n\, dS - \int_{\text{left}} \rho^n u^{n+1} u^n\, dS$$
$$+ \int_{\text{up}} \rho^n u^{n+1} v^n\, dS - \int_{\text{down}} \rho^n u^{n+1} v^n\, dS$$

The missing values of the Eulerian velocities and densities, at the sides of the control volume are interpolated as follows:

$$\int_{\text{right}} \rho^n u^{n+1} u^n dS \approx \rho^n_{i,j} \frac{u^{n+1}_{i,j}+u^{n+1}_{i+1,j}}{2}\frac{u^n_{i,j}+u^n_{i+1,j}}{2}\Delta y\,,$$
$$\int_{left} \rho^n u^{n+1} u^n dS \approx \rho^n_{i-1,j} \frac{u^{n+1}_{i,j}+u^{n+1}_{i-1,j}}{2}\frac{u^n_{i,j}+u^n_{i-1,j}}{2}\Delta y\,,$$
$$\int_{up} \rho^n u^{n+1} v^n dS \approx \rho^{n,up}_{i,j} \frac{u^{n+1}_{i,j}+u^{n+1}_{i,j+1}}{2}\frac{v^n_{i-1,j+1}+v^n_{i,j+1}}{2}\Delta x\,,$$
$$\int_{\text{down}} \rho^n u^{n+1} v^n dS \approx \rho^{n,down}_{i,j} \frac{u^{n+1}_{i,j}+u^{n+1}_{i,j-1}}{2}\frac{v^n_{i-1,j}+v^n_{i,j}}{2}\Delta x\,,$$

where

$$\rho_{i,j}^{n,up} = \frac{\rho_{i,j}^{n+1} + \rho_{i-1,j}^{n+1} + \rho_{i,j+1}^{n+1} + \rho_{i-1,j+1}^{n+1}}{4} \quad \text{and} \quad \rho_{i,j}^{n,down} = \frac{\rho_{i,j}^{n+1} + \rho_{i-1,j}^{n+1} + \rho_{i,j-1}^{n+1} + \rho_{i-1,j1}^{n+1}}{4}.$$

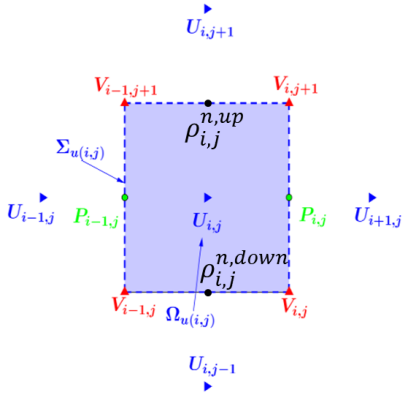


Figure 2.7: Control volume for the velocity component $u_{i,j}$. Pressure nodes $p_{i-1,j}$ and $p_{i,j}$ are located on the left and right sides of the rectangle, where the phase function C and the corresponding fluid densities $\rho_{i-1,j}$ and $\rho_{i,j}$ are defined. The densities $\rho_{i,j}^{n,\text{down}}$ and $\rho_{i,j}^{n,\text{up}}$ are associated with the bottom and top sides, respectively. The $v_{i,j}$ velocity components are positioned at the corners of the rectangle.

For the second term, we approximate it in order to apply the Divergence theorem, as follows:

$$\int_{\Omega_{u_{(i,j)}}} u^{n+1} \nabla \cdot (\rho^n \mathbf{u}^n)\, dV \approx u_{i,j}^{n+1} \int_{\Omega_{u_{(i,j)}}} \nabla \cdot (\rho^n \mathbf{u}^n)\, dV = u_{i,j}^{n+1} \int_{\Sigma_{u_{(i,j)}}} (\rho^n \mathbf{u}^n) \cdot \bar{n} dV$$

Now, following the similar procedure explained for the first term above we get:

$$\begin{aligned}
\int_{\text{right}} \rho^n u^n dS &\approx \rho_{i,j}^n \frac{u_{i,j}^n + u_{i+1,j}^n}{2} \Delta y\,, \\
\int_{left} \rho^n u^n dS &\approx \rho_{i-1,j}^n \frac{u_{i,j}^n + u_{i-1,j}^n}{2} \Delta y\,, \\
\int_{up} \rho^n v^n dS &\approx \rho_{i,j}^{n,up} \frac{v_{i-1,j+1}^n + v_{i,j+1}^n}{2} \Delta x\,, \\
\int_{down} \rho^n v^n dS &\approx \rho_{i,j}^{n,down} \frac{v_{i-1,j}^n + v_{i,j}^n}{2} \Delta x\,,
\end{aligned}$$

where

$$\rho_{i,j}^{n,up} = \frac{\rho_{i,j}^{n} + \rho_{i-1,j}^{n} + \rho_{i,j+1}^{n} + \rho_{i-1,j+1}^{n}}{4} \quad \text{and} \quad \rho_{i,j}^{n,down} = \frac{\rho_{i,j}^{n} + \rho_{i-1,j}^{n} + \rho_{i,j-1}^{n} + \rho_{i-1,j-1}^{n}}{4}.$$

Viscous terms

Let us recall, the strain tensor for two dimensions is given by

$$\left(\nabla u^{n+1} + \nabla^t u^{n+1}\right) = \kappa \begin{pmatrix} \frac{\partial u}{\partial x} & 0 \\ 0 & \frac{\partial v}{\partial y} \end{pmatrix} + \zeta \begin{pmatrix} 0 & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & 0 \end{pmatrix} - \eta \begin{pmatrix} 0 & \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \\ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} & 0 \end{pmatrix} \tag{2.23}$$

with $\kappa = 2\mu$ as the tearing viscosity, $\zeta = 2\mu$ as the shear viscosity, and $\eta = \mu$ as the rotational viscosity. Using the expression (2.23) in its surface integral form given by (2.21), we will separately consider the discretization of the integral for each of the three components that constitute viscosity.

1. The tearing viscosity term is written as

$$\begin{aligned}
\int_{\Omega_{u_{(i,j)}}} \frac{\partial}{\partial x}\left(\kappa \frac{\partial u}{\partial x}\right) dV = \int_{\Sigma_{u_{(i,j)}}} \left(\kappa \frac{\partial u}{\partial x}\right) dS &\approx \left[\kappa \frac{\partial u}{\partial x} \Delta y\right]_{b_x}^{f_x} \\
&\approx \left(\kappa_{i,j} \frac{u_{i+1,j}^{n+1} - u_{i,j}^{n+1}}{\Delta x} - \kappa_{i-1,j} \frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x}\right) \Delta y,
\end{aligned}$$

where b_x (f_x) denotes the back (front) face in the x direction.

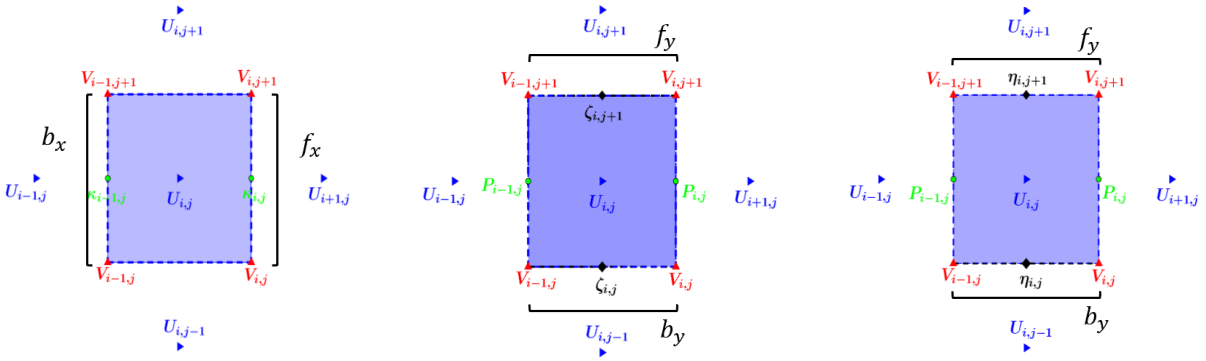
2. The shearing viscosity term is expressed as

$$\int_{\Omega_{u_{(i,j)}}} \frac{\partial}{\partial y}\left(\zeta\frac{\partial u}{\partial y}\right) dV = \int_{\Sigma_u}\left(\zeta\frac{\partial u}{\partial y}\right) dS \approx \left[\zeta\frac{\partial u}{\partial y}\Delta x\right]_{b_y}^{f_y}$$
$$\approx \left(\zeta_{i,j+1}\frac{u^{n+1}_{i,j+1}-u^{n+1}_{i,j}}{\Delta y} - \zeta_{i,j}\frac{u^{n+1}_{i,j}-u^{n+1}_{i,j-1}}{\Delta y}\right)\Delta x,$$

where b_y (f_y) denotes the back (front) face in the y direction.

3. The rotation term is written as

$$\int_{\Sigma_{u(i,j)}} \frac{\partial}{\partial y}\left(-\eta\frac{\partial u}{\partial y}+\eta\frac{\partial v}{\partial x}\right) dV \approx \left[-\eta\frac{\partial u}{\partial y}+\eta\frac{\partial v}{\partial x}\right]_{b_y}^{f_y}\Delta x$$
$$\approx \left(-\eta_{i,j+1}\frac{u^{n+1}_{i,j+1}-u^{n+1}_{i,j}}{\Delta y}+\eta_{i,j+1}\frac{v^{n+1}_{i,j+1}-v^{n+1}_{i-1,j+1}}{\Delta x}\right)\Delta x$$
$$+\left(\eta_{i,j}\frac{u^{n+1}_{i,j}-u^{n+1}_{i,j-1}}{\Delta y}-\eta_{i,j}\frac{v^{n+1}_{i,j}-v^{n+1}_{i-1,j}}{\Delta x}\right)\Delta x.$$



(a) (b) (c)

Figure 2.8: Control volume for the horizontal velocity component u, illustrating the locations of tearing viscosity (a), shearing viscosity (b), and rotational viscosity (c). The back and front faces in the x and y directions are also depicted, based on the requirements for discretizing the integrals of each viscosity component. The locations of the values for each component of the Eulerian velocity are also shown.

Calculation of the pressure gradient and velocity divergence terms

To calculate the pressure gradient, we follow a similar procedure as for the tearing viscosity term, using diagram (a) of Figure 2.8.

$$\int_{\Sigma_{u_{(i,j)}}} p^{n+1} n_x\, dS \approx \left[p^{n+1}\Delta y\right]_{b_x}^{f_x} \approx \left(p^{n+1}_{i,j} - p^{n+1}_{i-1,j}\right)\Delta y\,.$$

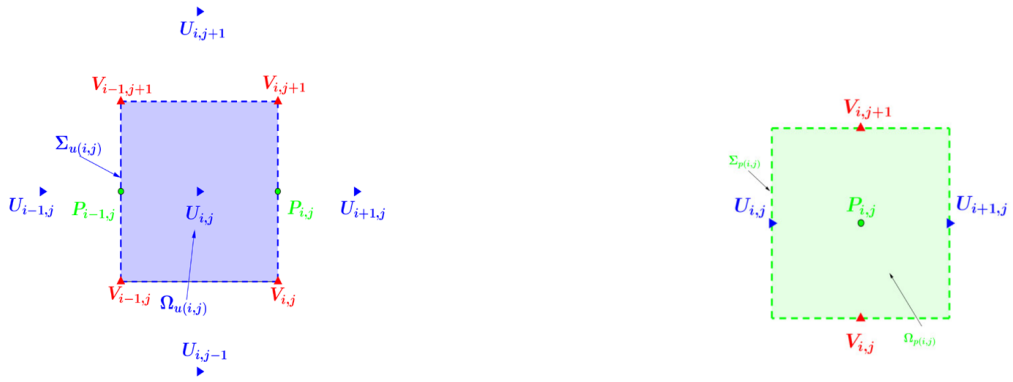
On the other hand, to calculate the velocity divergence $\mathbf{u}^{n+1} = (u^{n+1}, v^{n+1})$, it is approximated at the center of the cells (Fig. 2.9, diagram (b)) by

$$\int_{\Omega_p} \nabla\cdot\mathbf{u}^{n+1} dV = \int_{\Sigma_p} \mathbf{u}^{n+1}\cdot\bar{n} dS \approx \left(u^{n+1}_{i+1,j} - u^{n+1}_{i,j}\right)\Delta y + \left(v^{n+1}_{i,j+1} - v^{n+1}_{i,j}\right)\Delta x\,.$$

Regarding the vertical velocity component, v, a similar procedure is followed as was done for the u component to calculate the values of each integral.

Boundary conditions

Another important aspect to consider in our fictitious domain modeling method is the treatment of boundary conditions, for which we have employed the penalization method [2]. This method provides a



(a) (b)

Figure 2.9: Control volume for the u velocity component, showing the locations of pressure and Eulerian velocity components (a). Control volume for pressure, showing the Eulerian velocity components (b).

way to impose various types of boundary conditions, such as Dirichlet or Neumann. The implementation of this method involves using a control term in the momentum balance equation:

$$NS\left(\mathbf{u}^{n+1}\right) + \overline{\overline{B}}_{\text{penu}} \cdot \left(f\left(\mathbf{u}^{n+1}\right) - \mathbf{u}_{\infty}\right) = 0$$

with

$$\overline{\overline{B}}_{penu} = \begin{pmatrix} \alpha_u & 0 \\ 0 & \alpha_v \end{pmatrix}$$

The second-order tensor $\overline{\overline{B}}_{\text{penu}}$ defines the boundary conditions [2], where its diagonal elements approach infinity at the boundaries and are zero within the fluid domain. Moreover, $f(\mathbf{u}^{n+1})$ represents a discrete function of $\mathbf{u}^{n+1}$ and $\mathbf{v}^{n+1}$, expressed as a linear combination of the resolved velocities $u_{i,j}^{n+1}$ and $v_{i,j}^{n+1}$:

$$f(u^{n+1}) = a_0 u_{i,j}^{n+1} + a_1 u_{i-1,j}^{n+1} + a_2 u_{i+1,j}^{n+1} + a_3 u_{i,j-1}^{n+1} + a_4 u_{i,j+1}^{n+1},$$

$$f(v^{n+1}) = a_0 v_{i,j}^{n+1} + a_1 v_{i-1,j}^{n+1} + a_2 v_{i+1,j}^{n+1} + a_3 v_{i,j-1}^{n+1} + a_4 v_{i,j+1}^{n+1}.$$

The implementation of Dirichlet boundary conditions, applied for instance to the left boundary (see Fig. 2.10), is controlled by the coefficients a_i, which are defined as follows: $a_0 = a_2 = \frac{1}{2}$ and $a_1 = a_3 = a_4 = 0$. Thus, the discrete momentum equation at the left boundary becomes:

$$NS\left(u_{i,j}^{n+1}\right) + \alpha_u \left(\frac{1}{2}u_{0,j}^{n+1} + \frac{1}{2}u_{1,j}^{n+1} - u_{\infty\frac{1}{2},j}\right) = 0$$

with $u_{\infty\frac{1}{2},j} = 0$. To impose homogeneous Neumann boundary conditions, the coefficients are set to $a_0 = -1$, $a_2 = 1$, and $a_1 = a_3 = a_4 = 0$. The resulting discrete equation is written as:

$$NS\left(u_{i,j}^{n+1}\right) + \alpha_u \left(-u_{0,j}^{n+1} + u_{1,j}^{n+1} - u_{\infty\frac{1}{2},j}\right) = 0,$$

with $u_{\infty\frac{1}{2},j} = 0$.
Similarly, this procedure can be extended to the other boundaries of our physical domain. The reader may refer to the thesis of Mohamed El Ouafa (2022) [62], where a table is provided with the various coefficient values corresponding to the function f for Dirichlet or Neumann boundary conditions.

2.2.3 Developping a bidimensional fully coupled solver

After the finite volume discretization carried out in the previous sections, we generally obtain a non-symmetric linear system as follows:

$$\begin{pmatrix} F_u & B_p^T \\ B_u & 0 \end{pmatrix} \begin{pmatrix} \mathbf{u}^{n+1} \\ p^{n+1} \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix} \tag{2.24}$$

where $F_u = M_u^{(\rho)} + N_u^{(\rho)} + L_u^{(\mu)}$, and:

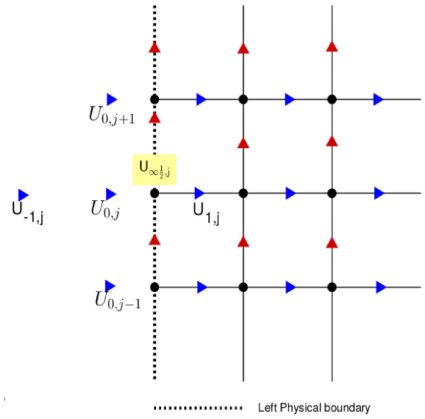


Figure 2.10: Treatment of boundary conditions on the left edge in 2D

- $B_u \mathbf{u}^{n+1} \approx \int_{\Omega_{i,j}} \nabla \cdot \mathbf{u}^{n+1}\, dV$ is the discrete divergence operator.
- $B_p^T p^{n+1} \approx \int_{\Omega_{i,j}} \nabla p^{n+1}\, dV$ is the discrete pressure gradient operator.
- $M_u^{(\rho)} \mathbf{u}^{n+1} \approx \int_{\Omega_{i,j}} \left(\frac{\rho^{n+1}}{\Delta t} \mathbf{u}^{n+1} + \overline{\overline{B}}_{\text{penu}} f\left(\mathbf{u}^{n+1}\right) \right) dV$ is the mass matrix.
- $L_u^{(\mu)} \mathbf{u}^{n+1} \approx -\int_{\Sigma_{i,j}} \left(\nabla \mathbf{u}^{n+1} + \nabla^t \mathbf{u}^{n+1} \right) \cdot \bar{n}\, dS$ is the viscous stress operator.
- $N_u^{(\rho)} \mathbf{u}^{n+1} \approx \int_{\Sigma_{i,j}} \rho \mathbf{u}^{n+1} \mathbf{u}^n \cdot \bar{n}\, dS - \int_{\Omega_{\mathbf{u}(i,j)}} \mathbf{u}^n \nabla \cdot \left(\rho^{n+1} \mathbf{u}^{n+1} \right) dV$ is the discrete convective term.
- $\mathbf{f} \approx \int_{\Omega_{i,j}} \left(\frac{\rho^n}{\Delta t} \mathbf{u}^n + \overline{\overline{B}}_{\text{penu}} \mathbf{u}_\infty + \mathbf{F}_{si} \right) dV$ is the right-hand side associated with velocity.

The system (2.24) can be rewritten in the reduced form $A\mathbf{x} = \mathbf{b}$, consisting of N equations with N unknowns and coefficients in $\mathbb{R}$. The matrix A is a square, sparse matrix with only 11 non-zero diagonals (see [62] for more details). Given this structure, iterative solvers are recommended as they are particularly effective in optimizing memory usage and reducing computational cost. For our specific problem, which involves flows laden with finite-sized particles, the BiCGSTAB(2) conjugate gradient method [25] has proven advantageous due to its strong performance in solving large, sparse, and nonsymmetric linear systems efficiently. Moreover, it is essential to combine the BiCGSTAB(2) algorithm with a matrix preconditioner to enhance the convergence of the iterative solver and mitigate potential numerical instabilities. In this case, the Pressure Convective-Diffusion (PCD) preconditioner was applied to the Schur component, while the block Gauss-Seidel preconditioner was used for the convection-diffusion-reaction operator (further details can be found in [62]).

2.3 Numerical treatment of particle's collisions

In this section, we will outline the process used to model particle-particle and particle-wall collisions, which is crucial in systems with both high and low solid fractions, such as those found in fluidized beds. This aspect is particularly important because it aims to achieve fully resolved numerical simulations and to prevent non-physical particle adhesion.

To manage particle interactions, a common method to prevent collisions is to define an exponential repulsive force [20] or a mass-damping force [34]. However, these methods do not account for the increased resistance that occurs when two particles, or a particle and a surface, come very close to each other due to the compression of a thin fluid film between them. This phenomenon, known as the lubrication effect, is crucial for modeling flow more realistically. To address lubrication, various approaches can be employed, including local mesh refinement in regions where lubrication effects need to be resolved. However, the main drawback of this approach is the high computational cost, especially in three-dimensional simulations. To address lubrication effects, various approaches exist. One common method involves applying local mesh refinement in regions where lubrication needs to be resolved. However, this approach is computationally expensive, especially in three dimensions. The approach used here, as proposed by Breugem (2010) [12], involves adding an equivalent repulsive force to compensate the unresolved hydrodynamic force when the distance between the two solids is less than an eulerian cell size. In the following two sections, we will briefly explain the treatment applied to address lubrication effects, as well as the approach taken for handling collisions.

2.3.1 Lubrication effect

When two solid surfaces are embedded in a viscous fluid, the lubrication effect generates a hydrodynamic force due to the drainage of the fluid. The following lubrication force for normal or head-on collisions is considered here:

$$\mathbf{F_l}\left(\epsilon_i, \mathbf{u}_n\right) = -6\pi\mu_f R\mathbf{u}_n\left[\lambda\left(\epsilon_i\right) - \lambda\left(\epsilon_{al}\right)\right] \tag{2.25}$$

with λ being defined as the amplification factor for the Stokes drag acting on an singles sphere is given in (2.26), (2.27) for particle-particle and particle-wall interaction, respectively:

$$\lambda_{pp} = \frac{1}{2\epsilon_i} - \frac{9}{20}\log\left(\epsilon_i\right) - \frac{3}{56}\epsilon_i\log\left(\epsilon_i\right) + 1.346 + \mathcal{O}(\epsilon_i) \tag{2.26}$$

$$\lambda_{pw} = \frac{1}{\epsilon_i} - \frac{1}{5}\log\left(\epsilon_i\right) - \frac{1}{21}\epsilon_i\log\left(\epsilon_i\right) + 0.9713 + \mathcal{O}(\epsilon_i), \tag{2.27}$$

where μ_f is the viscosity of the fluid, $\mathbf{u}_n$ is the normal velocity between the particles interaction, and $\epsilon_i = \frac{\|\mathbf{d}\|}{R}$ is the dimensionless distance between them. The activation distance ϵ_{al} is given by numerical assumptions explained in [11]. Breugem (2010) [12] extends the force $\mathbf{F_l}$ in 2.25 to account for lubrication corrections in non-normal collisions or with rotating particles.

The approach used to address lubrication effects is illustrated through a scenario where a spherical particle approaches a wall, as shown in Figure 2.11. The upward arrow indicates the distance between the particle and the wall, highlighting two critical points. The first, ϵ_{al}, corresponds to the mesh cell size, where the lubrication force rapidly increases due to λ. To control this force, a constant distance ϵ_1, representing the particle's roughness, is imposed. Below ϵ_1, the lubrication force is saturated until the particles overlap. It is at this point that the collision force is activated, which is based on a damping-mass force that effectively simulates the rebound effect during collisions (see section § 2.3.2).

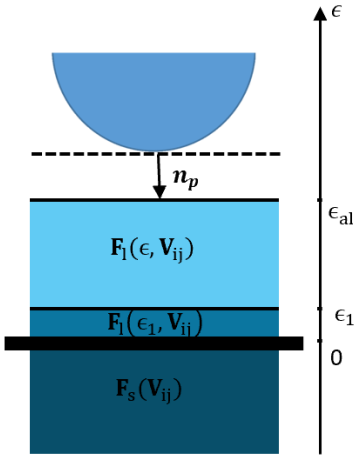


Figure 2.11: Multilayer model for hydrodynamic interaction for a head-on collision of a spherical particle and a wall.

2.3.2 Solid-solid collision

Previously, we discussed how the numerical treatment of lubrication forces can cause particles to overlap when they come very close, which does not realistically represent their behavior as some deformation is expected. To more accurately capture the bouncing effect that occurs between particles upon collision, it is important to consider the physical properties of the collision. It is known that a normal collision between two solids is characterized by the dry restitution coefficient e_d, which represents the amount of energy absorbed during the collision and is defined as the ratio of the velocity after to the velocity before the collision. However, when simulating collisions, one must consider that the duration of a collision, for instance between two approaching particles, can be as short as 10^{-9} seconds. Given that the time step in simulations of particle interactions within a flow (such as sedimentation flow or fluidized beds) typically ranges from 10^{-6} to 10^{-2} seconds, it is not feasible to directly simulate these collisions. As a solution, a repulsive force is often considered to prevent particle overlapping. This method is referred as the soft-sphere model and its major drawback is that it separates artificially the two interacting surfaces as soon as they are distant from one grid cell. Hence the collision effect is not accurate modeled and the restitution coefficient is generally underestimated [21]. In the work of Vincent et al. (2013) [79],

a standard approach to smooth spheres is employed by implementing a damping-mass force in place of the conventional repulsive force. This method explicitly incorporates the restitution coefficient, with the damping-mass force defined as follows:

$$\mathbf{F_s} = -k_n\delta - \beta_n u_n,$$
$$k_n = -\frac{m_e\left(\pi^2 + [\mathrm{Ln}e_d]^2\right)}{[N_c\Delta t]^2}, \beta_n = -\frac{2m_e\,[\mathrm{Ln}e_d]}{[N_c\Delta t]},$$

where k_n is the stiffness of the spring, β_n is the damping coefficient of the dashpot, and m_e characterizes the mass of solids involved in the collision. For solid-solid interaction, $m_e = \left(m_1^{-1} + m_2^{-1}\right)^{-1}$, and for particle-wall interaction, $m_e = m$. The numerical collision duration corresponds to $N_c\Delta t$, which is generally overestimated. However, a good compromise for achieving accurate collision results is to use $N_c = 8$. More details on the numerical implementations can be found in [11].

In the literature, various approaches have been proposed for modeling collisions. The distinctiveness of the method presented here lies in its implicit handling of collision forces ($\mathbf{F_s}$ and $\mathbf{F_l}$), which are directly incorporated into the momentum equation rather than being addressed during particle tracking. These forces are integrated into the momentum equation (2.2) by projecting the Lagrangian collision force of particle I ($\mathbf{F^I_{si}} = \mathbf{F_s} + \mathbf{F_l}$) as a volumetric force. The Lagrangian forces for a given particle are then distributed across the Eulerian cells within the particle as

$$\mathbf{F}_{si}(M) = \mathbf{F}^I_{si}/\left(V_p\right),$$

where M denotes an Eulerian grid point within particle I.

In the study of tribocharging in gas-solid flows, we aim to perform particle-resolved direct numerical simulations (PR-DNS) at the microscopic level, as we seek to track the charge distribution on the particle surface. At this scale, lubrication effects can become significant, and the treatment of such forces presented in section § 2.3.1 is insufficient due to saturation below ϵ_1 (see Fig. 2.11), which leads to an underestimation of the force and causes particle overlap. An alternative approach involves applying local grid refinement in the collision region, as proposed by Hu (1996) [39]. In the literature, several solvers with adaptive mesh refinement (AMR) have successfully been implemented, for instance, in the study of atmospheric boundary layers [77], multiphase flows [31], [54], and flows in complex geometries [65], [32]. A more recent work by Selçuk et al. (2021) [71] implemented AMR for particle-laden flows using the fictitious domain approach, similar to our case, and achieved successful validation even for challenging problems dominated by lubrication forces or involving non-spherical particles. Therefore, in the next chapter, we will provide an overview of the AMR method and discuss some results from its implementation.

Chapter 3

Adaptive mesh refinement

In numerical simulations, achieving high accuracy while maintaining manageable computational costs is a significant challenge, especially when dealing with complex phenomena that require varying levels of detail across different regions of the computational domain. Traditional methods, which apply a uniform grid resolution, often lead to inefficiencies, either by over-resolving areas of lesser importance or under-resolving critical regions. To address this, Adaptive Mesh Refinement (AMR) technology represents a powerful tool for locally enhancing the resolution of a grid, optimizing computational resources while maintaining accuracy. This approach is particularly useful in fields such as astrophysics, geophysics, aerospace engineering, and rail transportation, as AMR enables the minimization of computational costs while accurately capturing critical phenomena of interest.

Research in Adaptive Mesh Refinement (AMR) technologies dates back to the 1980s, with the pioneering work of Berger and Oliger, along with Collela [6, 7, 8], who initially introduced a dynamic mesh refinement method for solving time-dependent hyperbolic partial differential equations (PDEs) across multiple spatial dimensions on a structured Cartesian grid. This approach, known as the patch-based method, involves the creation of regular patches that are superimposed to refine the grid in regions of the domain where resolution is insufficient. The hierarchy of patches in this method consists of overlapping patches that are progressively nested as the level of refinement increases. In terms of implementation, a tree data structure was employed, which simplifies the management of the grid and the solution data generated adaptively [5]. Currently, software libraries utilizing the patch-based approach include Chombo [19], SAMRAI [82], and Enzo [64].

Following this, two main approaches were developed: Block-Structured AMR and Cell-Based AMR. The Block-Structured AMR approach, which can be seen as a simplified variant of the patched-based approach, was initially proposed by Berger [9] and Quirk [68], among others. In this approach, mesh adaptation is achieved by dividing the entire mesh into multiple blocks, each containing an identical number of grid cells. The refinement process is applied to these blocks, with data management facilitated by quadtree/octree data structures. Examples of libraries that implement this method include AMReX [87] and BoxLib [3].

On the other hand, the Cell-Based approach, introduced by Young et al.[85], also utilizes quadtree/octree data structures but differs in that the adaptive mesh refinement process is applied selectively to individual grid cells. This method allows for greater flexibility and precision by refining only the regions where high resolution is necessary, such as near discontinuities or steep gradients. Unlike patch-based or block-based methods that refine larger grid regions, the cell-based approach reduces computational costs and memory usage by minimizing unnecessary refinement. Among the libraries that implement this method are Basilisk and ART. Among the libraries that implement this method are Basilisk [66] and ART [86].

In this work, we employ a simplified version of the patch-based approach, focusing on the one-dimensional case, to outline the different stages involved in the adaptive meshing process. Our methodology is primarily based on the works of Berger and Oliger (1982) [8] and Berger and Collela (1989) [7], specifically considering the use of explicit finite difference schemes. Then we present some examples for the AMR.

3.1 Algorithm description

In this section, we present the main components of our algorithm, which is based on a patch-based approach, and detail certain simplifications specific to our one-dimensional AMR algorithm. Here, we consider Python as the programming language for implementing the various algorithms that construct the AMR framework. To access the code, please visit the following link: Github repository.

3.1.1 Grid description

Adaptive Mesh Refinement (AMR) relies on the use of hierarchically nested grids for the discretization of partial differential equations (PDEs). The process starts with the coarsest or base grid, G_0, which covers the entire computational domain and has a mesh spacing denoted by h_0. This base grid, G_0, may consist of a union of several overlapping rectangular grids, labeled as $G_{0,j}$, where $j \in I$. Generally, a grid is defined as the convex hull of its set of grid points, rather than just the grid points themselves. Consequently, the intersection of two grids is determined by the intersection of their convex hulls. Moreover, one grid is said to be contained within another if all its points lie within the convex hull that defines the other grid.

During the algorithm's execution, refined subgrids are adaptively generated in response to specific features of the evolving solution, such as the formation of shock fronts or based on error estimates. This generation process can be recursive, allowing refined subgrids to contain additional levels of refinement within their boundaries. At this point, it is important to highlight that each grid is not patched into the coarse grid but rather overlays it. Additionally, each grid is treated independently of the others, with its own solution, storage, and other resources. As a result, each subgrid can be integrated almost independently of the others, except at their boundaries. This approach also enables a flexible implementation of dynamic subgrids, even when the coarsest grid remains unchanged.

To establish a hierarchical distinction among subgrids based on their refinement level, we define the concept of a grid level, which indicates the number of coarser grids in which a subgrid is nested. Thus, the coarsest grid, G_0, is at level 0 in the hierarchy. Grids nested within G_0 form G_1 and are considered to be at the first level of refinement. Similarly, subgrids within G_1 belong to G_2, and this pattern continues. In this way, a sequence of nested grids with increasingly finer discretizations can be found within a portion of the spatial domain. In practice, to obtain a finer grid, the mesh size h_l, $l = 0, \ldots, l_{\max} - 1$ of a coarse grid is made a multiple of the mesh size h_{l+1} of the subgrid, such that $h_l = r h_{l+1}$, where r is known as the refinement ratio.

In the one dimensional case, the relationship between grids created at different refinement levels must be "properly nested", which means that:

(i) A finer grid must start and end at the sides of a cell on the next coarser grid level.

(ii) there must be at least one level $l-1$ cell within a level $l-1$ grid that separates a level l grid cell from a level $l-2$ grid cell in the left, and right directions, unless the cell is adjacent to the domain's physical boundary.

It is important to emphasize that, in the one-dimensional case, when considering more than two refined grids at the same level, these grids do not overlap. Instead, there are two possible scenarios: they either connect to form a larger grid, or they remain disjoint. For simplicity, in our approach, we do not account for the process of splitting grids; instead, each level of refinement is composed of only a single grid.

In higher dimensions, the requirement for proper nesting significantly increases the complexity of generating new refined grids. This complexity arises because a refined grid may be partially enclosed by two or more different coarser grids (see Fig. 3.1). For a more detailed discussion on this topic, the reader is referred to the work of Berger & Oliger [8].

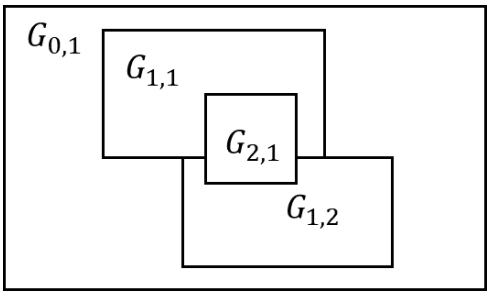


Figure 3.1: Example showing a fine grid correctly nested across two coarser grids [7].

3.1.2 Integration Algorithm

In this section, we present the integration algorithm used to solve hyperbolic PDEs. As outlined by Berger and Oliger [8], the algorithm consists of three main components, which are described below:

(i) The time-stepping process, which employs finite difference methods on each grid.

(ii) Error estimation followed by the generation of new grids.

(iii) Specialized inter-grid operations required at each time step during integration, driven by the mesh refinement process.

A significant portion of the computational effort is devoted to the time integration of the solution. Each grid is treated as an independent entity and evolves separately from the others, except for the determination of boundary values. This raises the natural question of how to order the integration of the grids. To address this, both spatial and temporal refinements are performed using the same refinement ratio. Specifically:

$$r = \frac{\Delta x_{l-1}}{\Delta x_l} = \frac{\Delta t_{l-1}}{\Delta t_l},\ l = 1, \ldots, l_{\max}.$$

This approach ensures that the mesh ratio, λ, which is the ratio between the time step and the spatial step, remains constant across all grids. Maintaining a consistent mesh ratio allows us to determine the order in which the grids should be integrated. For a refinement ratio r, each time step taken on a level 0 grid requires r time steps for level 1 grids, r^2 time steps for level 2 grids, and so on. These steps are interleaved such that, before advancing a grid to its next time step, all of its subgrids must be fully integrated up to the same current time level. This is crucial for maintaining consistent solution information across different grid levels. Furthermore, the fact that the smaller time step of the finer grids is not applied to the entire domain contributes to the efficiency of the AMR method.

Figure 3.2 illustrates the integration order in a one-dimensional case, assuming a refinement ratio of $r = 2$. The figure shows a grid at each level of refinement, with the corresponding time step sizes.

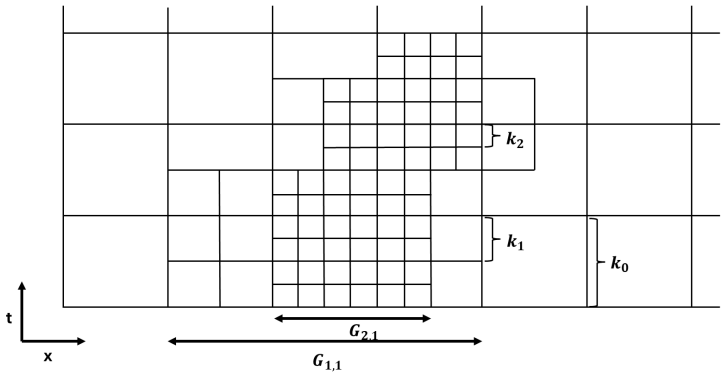


Figure 3.2: Example illustrating the development of spatial and temporal refinement in a one-dimensional case with a refinement ratio of 2. One grid is shown for each level of refinement, with the corresponding time step sizes [5].

One of the most critical components of the AMR method is error estimation and the subsequent regridding process, which is where the majority of the computational load occurs [5]. This component involves estimating the error at all grid points after a certain number of time steps and then adjusting the mesh structure accordingly. When a new refined grid is created, its values are initialized through linear or quadratic interpolation from the coarser grids [10]. Conversely, removing a grid simply involves reclaiming its storage. An important consideration for hyperbolic equations is the addition of a buffer zone around refined subgrids to extend the time between regridding processes. However, this involves a trade-off: a larger buffer zone reduces the frequency of regridding but requires more computational work to integrate the extra points within the buffer zone.

Finally, another important aspect to address involves communication routines between grids. Here, we consider two main tasks. The first involves managing the boundary conditions for the refined grids within the interior of the domain G_0. The boundary values are determined using the solution values from the underlying coarse grids through temporal interpolation to compute the boundary values at the times of the subgrids. The second task is updating the coarse grid, which is done by injecting the fine grid solution values onto the coarse grid points whenever a fine grid is nested within a coarse grid and both are integrated up to the same point in time. This approach is necessary for several reasons. First, if the coarse grid is not updated, the solution can become so diffused that, over time, further refinement may appear unnecessary. Second, by keeping a sharp discontinuity on the coarse grid, the region requiring refinement can be reduced in size. Lastly, this prevents a chain of incorrect values on the coarse grid from spreading into the buffer zone and contaminating the values that will be used for the boundary approximation on the fine grid [5].

The mesh refinement algorithm, in its entirety, is outlined in Fig. 1 It can be effectively formulated as a straightforward recursive procedure.

3.1.2.1 Grid generation

The grid generation process in Adaptive Mesh Refinement (AMR) involves identifying grid points that require refinement based on error estimation or other criteria. Once identified, new grid levels are created with a specific refinement ratio to enhance resolution in critical areas. To prevent excessive computational cost and complexity, a maximum level of refinement is set. This limitation ensures that the method balances accuracy and efficiency, avoiding overly refined grids that could lead to increased computational demands without proportional gains in precision.

To describe the grid generation more precisely, suppose we are at grid level l. The error is estimated at each point on the grid, and those points where the error (or another criterion for determining the need for refinement) exceeds a threshold ϵ are flagged. A subgrid with mesh width h_{l+1} is then created to encompass all the flagged points.

In general, when it is necessary to regrid, a new grid level may be introduced, an existing level may be updated, or an unnecessary level may be removed—even if a grid simply needs to be translated. In such cases, a refined grid is created and initialized with values from the old grid before it is discarded.

For our specific implementation, the regridding method involves identifying the flagged points and then considering the leftmost and rightmost points to create a new grid. It is important to note that, in our case, we have not considered the creation of multiple subgrids at the same level of refinement. However, to create different grids, we look for a sufficiently large gap between two flagged points, and this distance is determined by the buffer zone.

Algorithm 1 AMR algorithm

```
1: procedure INTEGRATE(lev)
2:     Repeat r_lev times
3:     if Regridding time? then
4:         Error estimation, grid generation,
5:         and creation of buffer zone
6:     end if
7:     Step Δt_lev on all grids at level lev
8:     if (lev + 1) grids exist then
9:         begin
10:        Integrate (lev + 1)
11:        Updating(lev, lev+1)
12:        end
13:    end if
14: end procedure
```

3.1.2.2 Data Structure

As outlined by Berger and Oliger [8], a data structure that tracks the relationships between different grids, as well as manages the storage of the solution, is crucial for the adaptive mesh refinement method. For our one-dimensional case, where each fine grid must be entirely contained within its coarser grid, the data structure used is a linked list. In this structure, each grid corresponds to a node, and there is a clear correspondence between the refined grid (child node) and its parent coarser grid. Consequently, all grid-to-grid operations, such as updating coarse grids from fine grids and setting internal boundary values for fine grids, follow an information flow that traverses the links in the linked list.

Additionally, because this tree-like structure can grow or shrink dynamically, some form of dynamic memory allocation is required, both for the grid information in each node and for the solution storage associated with each grid.

A grid is primarily characterized by the following pieces of information stored in each node of the linked list:

1. Level in the linked list.
2. Offspring pointer (if it exists).
3. Parent pointer.
4. Time up to which this grid has been integrated.
5. Solution vector.

6. Spatial discretization.

In the more general case, where multiple subgrids may exist at the same level of refinement, the data structure becomes more complex. For such cases, a tree structure is more appropriate, as it requires additional connections between grids originating from the same parent (referred to as siblings) and between grids from different parents (referred to as neighbors). More information on this can be found in [8].

3.1.3 Error estimator

The error estimation method we employ relies on calculating the local truncation error through Richardson extrapolation. To introduce this error estimator, we first need to provide some definitions, as outlined by Oliger & Zhu (1996) [61].

Consider the following problem:

$$u_t = Lu + f, \qquad \text{on } \Omega \times [0,T], \tag{3.1}$$
$$u(0) = u_0, \qquad \text{on } \Omega, \tag{3.2}$$
$$Bu = b, \qquad \text{on } \partial\Omega \times [0,T], \tag{3.3}$$

where $\Omega \subset \mathbb{R}^d$ is a bounded domain in physical space, L is a spatial partial differential operator defined on Ω, and $u \in \mathbb{R}^n$ represents the solution function, assumed to form a well-posed initial-boundary value problem. The discretizations of Ω, $\partial\Omega$, and $[0,T]$ are given by Ω_h, $\partial\Omega_h$, and $[0,T]_k$, respectively, with h and k denoting the spatial and temporal step sizes.

Let v_h be a grid function defined on $\Omega_h \times [0,T]_k$. To maintain generality and simplicity while avoiding cumbersome notation, we express our methods in an explicit one-step format as follows:

$$v_h(t+k) = v_h(t) + k(L_h v_h(t) + f_h(t)) \quad \text{on } \Omega_h \times [0,T]_k, \tag{3.4}$$
$$v_h(0) = u_{0,h} \quad \text{on } \Omega_h, \tag{3.5}$$

where subscripts indicate the projection of functions onto the corresponding grids, as well as the discretization of operators on these grids. If u_h denotes the projection of the exact solution of the system onto Ω_h, then

$$u_h(t+k) = u_h(t) + kL_h u_h(t) + kf_h(t) + k\tau_h, \quad \text{on } \Omega_h \times [0,T]_k,$$

where $k\tau_h$ represents the local truncation error. If we denote by Q_h the one-step scheme given by (3.4), the local truncation error can be written as (assuming the solution is sufficiently smooth):

$$\begin{aligned} k\tau_h(x,t) &= u_h(x,t+k) - Q_h u_h(x,t) \\ &= k\left(k^{p_1} a(x,t) + h^{p_2} b(x,t)\right) + kO\left(k^{p_1+1} + h^{p_2+1}\right) \\ &= k\tau + kO\left(k^{p_1+1} + h^{p_2+1}\right). \end{aligned} \tag{3.6}$$

The leading term is denoted by $k\tau$. If u is sufficiently smooth and we take two time steps with the operator Q_h, then, to leading order, the error is $2k\tau$:

$$u_h(x,t+2k) - Q_h^2 u_h(x,t) = 2k\tau + kO(k^{p_1+1} + h^{p_2+1}). \tag{3.7}$$

Let Q_{2h} be the same difference operator as Q_h, but based on grids with widths $2h$ and $2k$. Assume also that the order of accuracy in time and space is the same, i.e., $p_1 = p_2 = p$. Then,

$$\begin{aligned} u_h(x,t+2k) - Q_{2h} u_h(x,t) &= 2k\left((2k)^p a(x,t) + (2h)^p b(x,t)\right) + O(h^{p+2}), \\ &= 2^{p+1} k\tau + O(h^{p+2}). \end{aligned} \tag{3.8}$$

Subtracting (3.7) from (3.8) and using the expression (3.6), we obtain

$$\tau_h(x,t) = \frac{Q_h^2 u_h(x,t) - Q_{2h} u_h(x,t)}{k(2^{p+1} - 2)} + O(h^{p+1}).$$

This provides an estimator of the local truncation error at time t. In words, one can estimate the pointwise error by taking a large step using mesh widths of $2h$ and $2k$, starting from the solution at time t. Then, this solution is compared to the one obtained by taking two regular integration steps (see Figure 3.3). Note that this error estimator is also valid for piecewise uniform grids, such as adaptively generated subgrids.

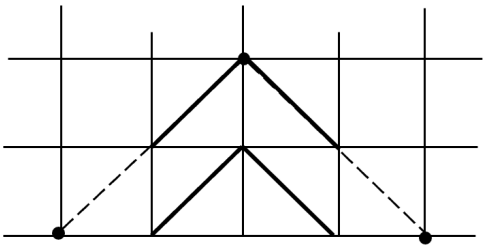


Figure 3.3: Richardson error estimation procedure [8].

The advantage of this procedure lies in the fact that it is not necessary to know the exact form of the truncation error to apply it, as, for example, one does not need to explicitly know the functions a and b that involve higher derivatives of the solution [8]. Furthermore, note that this estimator is independent of the finite difference scheme used.

The restriction that the accuracy in time and space be the same is not a serious limitation. For more details, refer to Berger [8]. For non-smooth solutions, we can no longer have an accurate error estimate. However, Richardson estimates still provide a good criterion for refinement, as the estimates will be large near a singularity.

3.2 Numerical examples

In order to show the results of our adaptive mesh refinement let us consider the well-known transport equation:

$$\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} = 0, \text{ where } x \in [e, d], t \in [0, +\infty[. \tag{3.9}$$

The unique solution, given by its initial condition $u_0(x) = u(x, 0)$, $x \in [e, d]$, is given as follows

$$u(x, t) = u_0(x - at). \tag{3.10}$$

Now, we will use the Adaptive Mesh Refinement (AMR) method for different initial conditions to study the performance of AMR by considering two schemes: the Lax-Wendroff scheme and the Forward in Time, Backward in Space (FTBS) scheme.

1. Let us consider the following Gaussian initial condition given by:

$$u_0(x) = 0.2\exp(-(x-4)^2) \tag{3.11}$$

   First, we consider a study of the convergence order for these two schemes (Lax-Wendroff anf FTBS) by varying the spatial resolution for different Courant numbers. This is because, for certain Courant number cases, the order is not as expected, and we aim to study the error of the AMR without interference from other factors. This gives us the following graph:

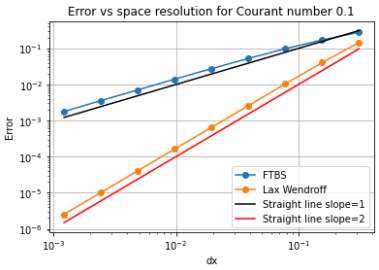
   We now present a comparison between the solution graphs obtained with and without Adaptive Mesh Refinement (AMR). For this analysis, we use initial parameters of $e = 0$ and $d = 10$ for the spatial domain limits, with 80 spatial points and a maximum time of 1.5. The results obtained under these conditions are as follows:

   We observe an improvement in the solution in the graph where AMR is applied (see Fig. 3.5a). Additionally, we note that the refinement region is quite large, as evidenced by the size of the subgrid at level 1. Our code does not yet account for cases with multiple subgrids. For other cases with different Courant numbers, we have chosen not to include their graphs, as the results are very similar. Now, let us consider the case where the Lax-Wendroff scheme is used.

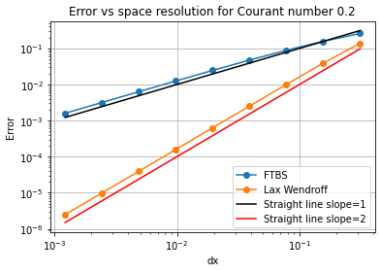
   As expected, since the Lax-Wendroff scheme is second-order accurate in both space and time, the solution is more precise than with the FTBS scheme, as shown in Fig. 3.6a. The same parameters were used as in the previous case.

2. For this case, let us consider an initial discontinuous solution of the form:

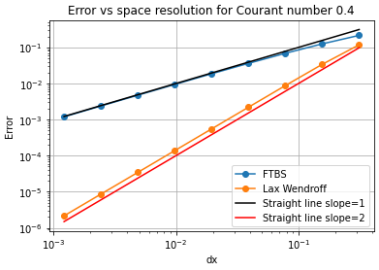
$$u_0(x) = \begin{cases} 1.0 & \text{if } x < 1.4, \\ 0 & \text{otherwise.} \end{cases}$$



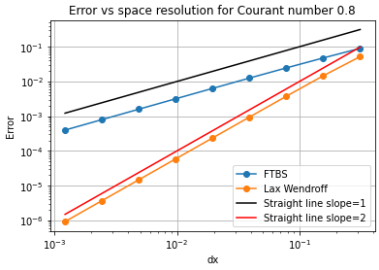
(a) Courant number c=0.1



(b) Courant number c=0.2

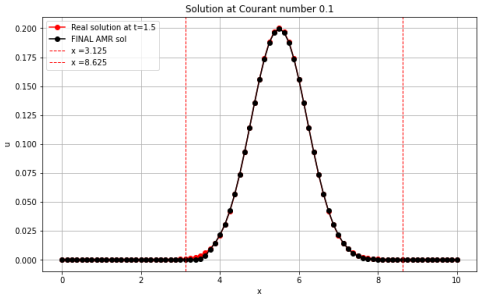


(c) Courant number c=0.4

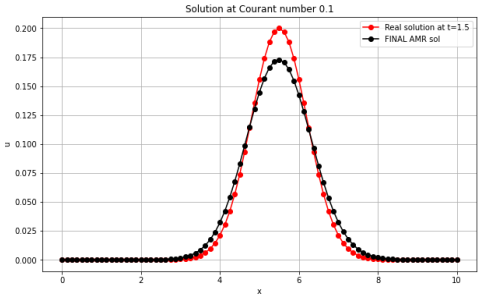


(d) Courant number c=0.8

Figure 3.4: Plots of convergence orders of schemes FTBS and Lax Wendroff for different values of Courant number



(a) AMR solution FTBS, ratio of refinement 8 and threshhold 10^{-6} and maximum level of refinement 2.



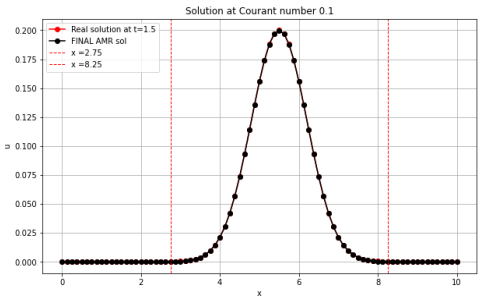
(b) Solution without AMR FTBS

Figure 3.5: Comparison of the approximated solution using the FTBS scheme with AMR (a) and without AMR (b), considering a Courant number of 0.1. The two dashed lines represent the refinement subgrid used to obtain the final solution.

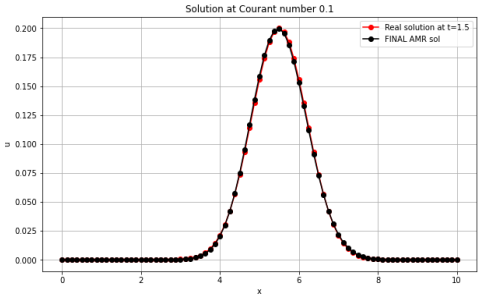
which gives us a Riemann problem. For this case, we will follow a similar procedure to that used for the previous initial condition and use the same parameters for spatial discretization and final time.

We will plot the convergence order for the cases with Courant numbers that best match the theoretical convergence order (see Fig. 3.7a and 3.7b). Therefore, the values considered will be $c = 0.1$ and $c = 0.2$.

In the graphs, we observe that as the spatial discretization resolution increases, the convergence order plot starts to resemble the theoretical convergence order. Next, we proceed to plot the

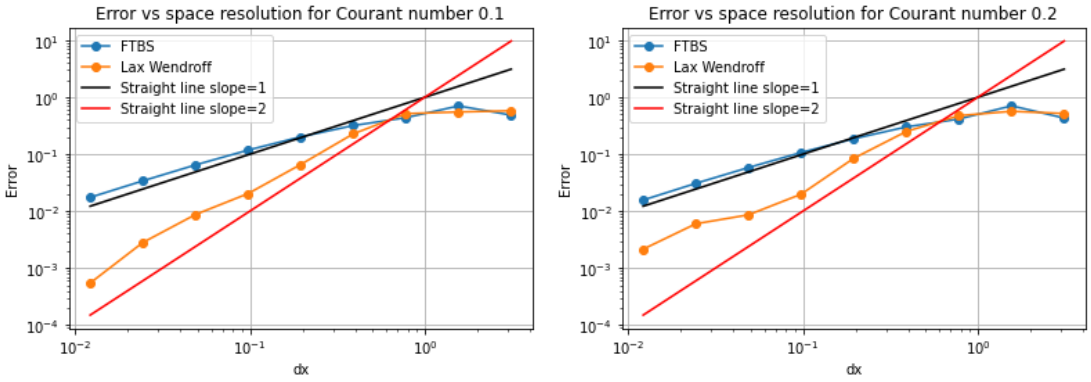


(a) AMR solution with L-W, ratio of refinement 8 and threshhold 10^{-6} and maximum level of refinement 2



(b) Solution without AMR L-W

Figure 3.6: Comparison of the approximated solution using the Law-Wendroff scheme with AMR (a) and without AMR (b), considering a Courant number of 0.1. The two dashed lines represent the refinement subgrid used to obtain the final solution.



(a) Courant number c=0.1

(b) Courant number c=0.2

Figure 3.7: Plots of convergence orders of schemes FTBS and Lax Wendroff for different values of Courant number

solutions with and without Adaptive Mesh Refinement (AMR) for both types of schemes.

Regarding the FTBS scheme (see Fig. 3.9a and Fig. 3.9b), a more accurate solution can be observed when using AMR. However, with a refinement ratio of 16 and a maximum of 2 refinement levels, we can still see that the solution is not exact in the region of the discontinuity. A similar result occurs for the Lax-Wendroff scheme (see Fig. 3.10a and 3.10b), where the exact solution exhibits an oscillatory behavior when dealing with initial conditions involving discontinuities. On the other hand, when using AMR, an improvement in the solution is observed, except in the region of the discontinuity, where a small oscillation can be noticed.

Since oscillations, as shown in Fig. 3.10a, as well as undesired displacements of the solution (see Fig. 3.8), have been encountered when considering different types of examples, we have not considered conducting an error analysis because our code still has deficiencies that have not yet been resolved. This deficiencies appeared for example when increasing the level of refinement and also the maximum level of refinement.

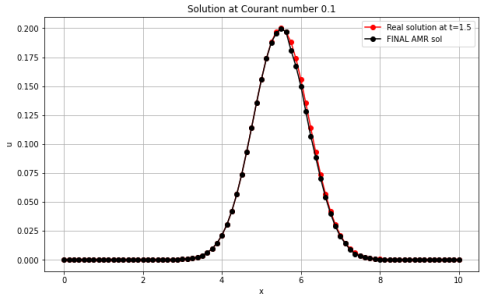
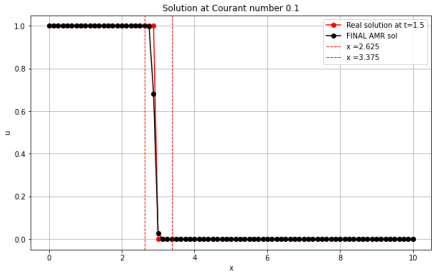
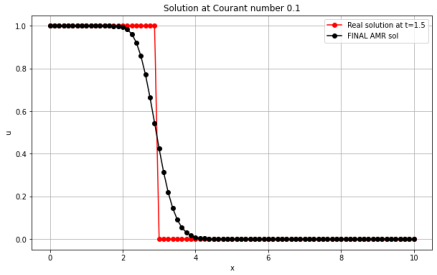


Figure 3.8: AMR solution for ratio of refinement 16, and maximum level of refinement 2.

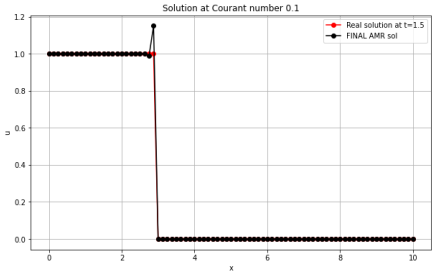


(a) AMR solution FTBS, ratio of refinement 16 and threshhold 10^{-6}.

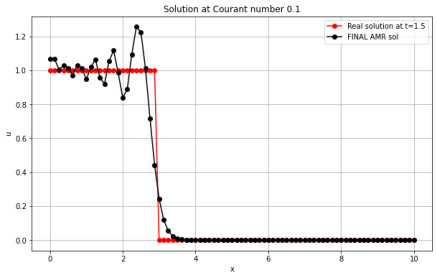


(b) Solution without AMR FTBS

Figure 3.9: Comparison of the approximated solution using the FTBS scheme with AMR (a) and without AMR (b), considering a Courant number of 0.1. The two dashed lines represent the refinement subgrid used to obtain the final solution.



(a) AMR solution with L-W, ratio of refinement 16 and threshhold 10^{-6}.



(b) Solution without AMR L-W

Figure 3.10: Comparison of the approximated solution using the Law-Wendroff scheme with AMR (a) and without AMR (b), considering a Courant number of 0.1. The two dashed lines represent the refinement subgrid used to obtain the final solution.

Conclusions

After a comprehensive literature review on Adaptive Mesh Refinement (AMR), we implemented an adaptive mesh refinement method for the one-dimensional case with specific simplifications following the work of Berger and Oliger (1982) [5]. This was done to demonstrate the practical functioning of this method and to explore potential improvements in handling collisions, particularly in the context of the viscous penalty method, which is known for its drawbacks in terms of computational cost and accuracy when applied to multiphase flows and other fluid dynamics problems.

Our results started with a thorough analysis of the convergence order to identify suitable Courant numbers, aiming to minimize errors inherent to the two numerical schemes employed: Lax-Wendroff and Forward in Time, Backward in Space (FTBS). This analysis was necessary because the choice of Courant number can significantly influence the stability and accuracy of numerical schemes, especially in the context of AMR where multiple grid levels and time steps interact. Two types of initial conditions were examined: one involving a Gaussian function and the other dealing with a discontinuity. Through these cases, we illustrated the capability of the AMR method to refine the grid dynamically in regions where higher resolution is needed, thus enhancing the accuracy of the approximate solution.

From the graphical results, it is evident that the AMR method significantly enhances the precision of the approximate solution by allocating computational resources to the most critical regions. However, some unexpected behaviors were observed, such as oscillations near discontinuities and shifts in the approximate solutions. Although the root causes of these issues are not fully understood at present, resolving them is a priority for future work. This will involve refining the method to improve its stability and accuracy, especially around regions of steep gradients or discontinuities.

Looking ahead, there are several possible extensions to this work. One promising direction is to include multiple subgrids at the same level of refinement. This would create a more sophisticated algorithm due to the challenges of grid overlapping and proper synchronization of boundary conditions between grids. Moreover, extending this approach to higher-dimensional problems could leverage tree data structures to handle the complexity of grid management more effectively. Another important extension would be to apply finite volume methods instead of finite difference methods for solving hyperbolic equations. This approach is generally more robust in handling conservation laws, especially in the presence of shocks or steep gradients, and could mitigate some of the issues observed with the current implementation.

Overall, the current work provides a foundational understanding and practical insights into AMR methods, but there is room for significant advancement, particularly in the areas of algorithmic complexity, grid management, and numerical stability. These improvements will be crucial for applying AMR to more complex, real-world problems, such as fluid dynamics in multiphase flows.

Bibliography

[1] F Sharmene Ali, M Adnan Ali, R Ayesha Ali, and Ion I Inculet. Minority charge separation in falling particles with bipolar charge. *Journal of Electrostatics*, 45(2):139–155, 1998.

[2] Philippe Angot, Charles-Henri Bruneau, and Pierre Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numerische Mathematik*, 81(4):497–520, 1999.

[3] J Bell, A Almgren, V Beckner, M Day, M Lijewski, A Nonaka, and W Zhang. Boxlib user's guide. *github. com/BoxLib-Codes/BoxLib*, 2012.

[4] M Beretta, TR Hörmann, P Hainz, W-K Hsiao, and A Paudel. Investigation into powder tribocharging of pharmaceuticals. part i: Process-induced charge via twin-screw feeding. *International Journal of Pharmaceutics*, 591:120014, 2020.

[5] Marsha J. Berger. Adaptive mesh refinement for hyperbolic partial differential equations. Technical Report STAN-CS-82-924, Department of Computer Science, Stanford University, Stanford, CA, August 1982. Approved for public release; distribution unlimited.

[6] Marsha J Berger. Data structures for adaptive grid generation. *SIAM Journal on Scientific and Statistical Computing*, 7(3):904–916, 1986.

[7] Marsha J Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of computational Physics*, 82(1):64–84, 1989.

[8] Marsha J Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of computational Physics*, 53(3):484–512, 1984.

[9] Marsha J Berger and Jeff S Saltzman. Amr on the cm-2. *Applied Numerical Mathematics*, 14(1-3):239–253, 1994.

[10] John H Bolstad. An adaptive finite difference method for hyperbolic systems in one space dimension. 1982.

[11] Jorge César Brändle De Motta, W-P Breugem, Bertrand Gazanion, J-L Estivalezes, Stéphane Vincent, and Éric Climent. Numerical modelling of finite-size particle collisions in a viscous fluid. *Physics of Fluids*, 25(8), 2013.

[12] Wim-Paul Breugem. A combined soft-sphere collision/immersed boundary method for resolved simulations of particulate flows. In *Fluids Engineering Division Summer Meeting*, volume 49484, pages 2381–2392, 2010.

[13] Carsten Burstedde, Lucas C Wilcox, and Omar Ghattas. p4est: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM Journal on Scientific Computing*, 33(3):1103–1133, 2011.

[14] Rubén M Cabezón, Domingo García-Senz, and Antonio Relaño. A one-parameter family of interpolating kernels for smoothed particle hydrodynamics studies. *Journal of Computational Physics*, 227(19):8523–8540, 2008.

[15] Jean-Paul Caltagirone and Stéphane Vincent. Sur une méthode de pénalisation tensorielle pour la résolution des équations de navier–stokes. *Comptes Rendus de l'Académie des Sciences-Series IIB-Mechanics*, 329(8):607–613, 2001.

[16] Mohamed-Amine Chadil, Stéphane Vincent, and Jean-Luc Estivalezes. Improvement of the viscous penalty method for particle-resolved simulations. *Open Journal of Fluid Dynamics*, 9(02):168–192, 2019.

[17] Fahad Al-Amin Chowdhury. *CFD Simulation of Electrostatic Charging in Gas-Solid Fluidized Beds: Model Development Through Fundamental Charge Transfer Experiments*. Doctor of philosophy in chemical engineering, University of Ottawa, Ottawa, Canada, 2021.

[18] J Ciborowski and A_ Wlodarski. On electrostatic effects in fluidized beds. *Chemical Engineering Science*, 17(1):23–32, 1962.

[19] Phillip Colella, Daniel T Graves, TJ Ligocki, DF Martin, D Modiano, DB Serafini, and B Van Straalen. Chombo software package for amr applications design document. *Available at the Chombo website: http://seesar. lbl. gov/ANAG/chombo/(September 2008)*, 2, 2009.

[20] Mathieu Coquerelle and G-H Cottet. A vortex level set method for the two-way coupling of an incompressible fluid with colliding rigid bodies. *Journal of Computational Physics*, 227(21):9121–9137, 2008.

[21] C Crowe, M Sommerfeld, Y Tsuji, and C Crowe. Multiphase flows with droplets and particles crc. *Boca Raton, FL*, 1998.

[22] DK Davies. Charge generation on dielectric surfaces. *Journal of Physics D: Applied Physics*, 2(11):1533, 1969.

[23] Jean-Marc Delhaye. Jump conditions and entropy sources in two-phase systems. local instant formulation. *International Journal of Multiphase Flow*, 1(3):395–409, 1974.

[24] Cécile Dobrzynski and Pascal Frey. Anisotropic delaunay mesh adaptation for unsteady simulations. In *Proceedings of the 17th international Meshing Roundtable*, pages 177–194. Springer, 2008.

[25] Jack J Dongarra, Iain S Duff, Danny C Sorensen, and Henk A Van der Vorst. *Numerical linear algebra for high-performance computers*. SIAM, 1998.

[26] Mohamed El Ouafa, Vincent Stéphane, and Vincent Le Chenadec. Navier-stokes solvers for incompressible single-and two-phase flows. *Communications in Computational Physics*, 2020.

[27] Akihiko Ema, Daisuke Yasuda, Ken-ichiro Tanoue, and Hiroaki Masuda. Tribo-charge and rebound characteristics of particles impact on inclined or rotating metal target. *Powder technology*, 135:2–13, 2003.

[28] Liang-Shih Fan. *Gas-liquid-solid fluidization engineering*. Butterworth-Heinemann, 2013.

[29] Keith M Forward, Daniel J Lacks, and R Mohan Sankaran. Methodology for studying particle–particle triboelectrification in granular materials. *Journal of Electrostatics*, 67(2-3):178–183, 2009.

[30] M Fujino, S Ogata, and H Shinohara. The electric potential distribution profile in a naturally charged fluidized bed and its effects. *Int. Chem. Eng.;(United States)*, 25(1), 1985.

[31] Daniel Fuster, Anne Bagué, Thomas Boeck, Luis Le Moyne, Anthony Leboissetier, Stéphane Popinet, Pascal Ray, Ruben Scardovelli, and Stéphane Zaleski. Simulation of primary atomization with an octree adaptive mesh refinement and vof method. *International Journal of Multiphase Flow*, 35(6):550–565, 2009.

[32] Frederic Gibou, Ronald P Fedkiw, Li-Tien Cheng, and Myungjoo Kang. A second-order-accurate symmetric discretization of the poisson equation on irregular domains. *Journal of Computational Physics*, 176(1):205–227, 2002.

[33] Dimitri Gidaspow. *Multiphase flow and fluidization: continuum and kinetic theory descriptions*. Academic press, 1994.

[34] Roland Glowinski, Tsorng-Whay Pan, Todd I Hesla, Daniel D Joseph, and Jácques Periaux. A fictitious domain approach to the direct numerical simulation of incompressible viscous flow past moving rigid bodies: application to particulate flow. *Journal of computational physics*, 169(2):363–426, 2001.

[35] Robert O Hagerty, Michael E Muhle, Agapios K Agapiou, Chi-I Kuo, Mark G Goode, F David Hussein, Richard B Pannell, and John F Szul. Method for controlling sheeting in gas phase reactors, July 26 2011. US Patent 7,985,811.

[36] John Happel and Howard Brenner. *Low Reynolds number hydrodynamics: with special applications to particulate media*, volume 1. Springer Science & Business Media, 2012.

[37] Wallace Russell Harper. Contact and frictional electrification. *(No Title)*, 1967.

[38] Gregory Hendrickson. Electrostatics and gas phase fluidized bed polymerization reactor wall sheeting. *Chemical Engineering Science*, 61(4):1041–1064, 2006.

[39] Howard H Hu. Direct simulation of flows of solid-liquid mixtures. *International Journal of Multiphase Flow*, 22(2):335–352, 1996.

[40] Howard H Hu, Neelesh A Patankar, and MY1836522 Zhu. Direct numerical simulations of fluid–solid systems using the arbitrary lagrangian–eulerian technique. *Journal of Computational Physics*, 169(2):427–462, 2001.

[41] Takayuki Itakura, Hiroaki Masuda, Chiyo Ohtsuka, and Shuji Matsusaka. The contact potential difference of powder and the tribo-charge. *Journal of Electrostatics*, 38(3):213–226, 1996.

[42] Paul Iversen and Daniel J Lacks. A life of its own: The tenuous connection between thales of miletus and the study of electrostatic charging. *Journal of Electrostatics*, 70(3):309–311, 2012.

[43] Isao Kataoka. Local instant formulation of two-phase flow. *International Journal of Multiphase Flow*, 12(5):745–758, 1986.

[44] Isao Kataoka, Mamoru Ishii, and Akimi Serizawa. Local formulation and measurements of interfacial area concentration in two-phase flow. *International Journal of Multiphase Flow*, 12(4):505–529, 1986.

[45] Tobias Kempe and Jochen Fröhlich. Collision modelling for the interface-resolved simulation of spherical particles in viscous fluids. *Journal of Fluid Mechanics*, 709:445–489, 2012.

[46] Khodor Khadra, Philippe Angot, Sacha Parneix, and Jean-Paul Caltagirone. Fictitious domain approach for numerical modelling of navier–stokes equations. *International journal for numerical methods in fluids*, 34(8):651–684, 2000.

[47] Daniel J Lacks and Troy Shinbrot. Long-standing and unresolved issues in triboelectric charging. *Nature Reviews Chemistry*, 3(8):465–476, 2019.

[48] JC Laurentie, P Traoré, and L Dascalescu. Discrete element modeling of triboelectric charging of insulating materials in vibrated granular beds. *Journal of Electrostatics*, 71(6):951–957, 2013.

[49] Chun Hean Lee, Antonio J Gil, Paulo R Refachinho de Campos, Javier Bonet, Tadas Jaugielavičius, Shreyas Joshi, and Clare Wood. A novel arbitrary lagrangian eulerian smooth particle hydrodynamics algorithm for nonlinear solid dynamics. *Computer Methods in Applied Mechanics and Engineering*, 427:117055, 2024.

[50] Victor Lee, Scott R Waitukaitis, Marc Z Miskin, and Heinrich M Jaeger. Direct observation of particle interactions and clustering in charged granular streams. *Nature Physics*, 11(9):733–737, 2015.

[51] Paola Lettieri and Luca Mazzei. Challenges and issues on the cfd modeling of fluidized beds: a review. *The Journal of Computational Multiphase Flows*, 1(2):83–131, 2009.

[52] Hongwei Li, Lei Wang, Changhe Du, and Wenpeng Hong. Cfd-dem investigation into flow characteristics in mixed pulsed fluidized bed under electrostatic effects. *Particuology*, 66:100–112, 2022.

[53] Yi Liang, Cheng Wang, and Pengtao Sun. An interface-fitted fictitious domain finite element method for the simulation of neutrally buoyant particles in plane shear flow. *Fluids*, 8(8):229, 2023.

[54] Frank Losasso, Frédéric Gibou, and Ron Fedkiw. Simulating water and smoke with an octree data structure. In *Acm siggraph 2004 papers*, pages 457–462. ACM Transactions on Graphics (TOG), 2004.

[55] Shuji Matsusaka, Mojtaba Ghadiri, and Hiroaki Masuda. Electrification of an elastic sphere by repeated impacts on a metal plate. *Journal of Physics D: Applied Physics*, 33(18):2311, 2000.

[56] Shuji Matsusaka, H Maruyama, Tatsushi Matsuyama, and Mojtaba Ghadiri. Triboelectric charging of powders: A review. *Chemical Engineering Science*, 65(22):5781–5807, 2010.

[57] Tatsushi Matsusyama and Hideo Yamamoto. Impact charging of particulate materials. *Chemical Engineering Science*, 61(7):2230–2238, 2006.

[58] Bertrand Maury. Direct simulations of 2d fluid-particle flows in biperiodic domains. *Journal of computational physics*, 156(2):325–351, 1999.

[59] Poupak Mehrani, Matti Murtomaa, and Daniel J Lacks. An overview of advances in understanding electrostatic charge buildup in gas-solid fluidized beds. *Journal of Electrostatics*, 87:64–78, 2017.

[60] AW Nienow. Fluidised bed granulation and coating: applications to materials, agriculture and biotechnology. *Chemical Engineering Communications*, 139(1):233–253, 1995.

[61] Joseph Oliger and Xiaolei Zhu. Stability and error estimation for component adaptive grid methods. *Applied numerical mathematics*, 20(4):407–426, 1996.

[62] Mohamed El Ouafa. *Développement d'un solveur tout-couplé parallèle 3D pour la simulation des écoulements diphasiques incompressibles à forts rapports de viscosités et de masses volumiques*. PhD thesis, Université Gustave Eiffel, Paris, France, December 2022. Thèse de doctorat en Mécanique, soutenue publiquement le 15 Décembre 2022.

[63] Bo Ouyang, Li-Tao Zhu, and Zheng-Hong Luo. Data-driven modeling of mesoscale solids stress closures for filtered two-fluid model in gas–particle flows. *AIChE Journal*, 67(7):e17290, 2021.

[64] Brian W O'shea, Greg Bryan, James Bordner, Michael L Norman, Tom Abel, Robert Harkness, and Alexei Kritsuk. Introducing enzo, an amr cosmology application. In *Adaptive Mesh Refinement-Theory and Applications: Proceedings of the Chicago Workshop on Adaptive Mesh Refinement Methods, Sept. 3–5, 2003*, pages 341–349. Springer, 2005.

[65] Stéphane Popinet. Gerris: a tree-based adaptive solver for the incompressible euler equations in complex geometries. *Journal of computational physics*, 190(2):572–600, 2003.

[66] Stéphane Popinet. Quadtree-adaptive tsunami modelling. *Ocean Dynamics*, 61:1261–1285, 2011.

[67] Constantine Pozrikidis. *Introduction to theoretical and computational fluid dynamics*. Oxford university press, 2011.

[68] James J Quirk and Ulf R Hanebutte. A parallel adaptive mesh refinement algorithm. Technical report, Institute for computer applications in science and engineering, 1993.

[69] Vivek V Ranade. *Computational flow modeling for chemical reactor engineering*. Elsevier, 2001.

[70] Stefanie Rauchenzauner and Simon Schneiderbauer. A dynamic anisotropic spatially-averaged two-fluid model for moderately dense gas-particle flows. *International Journal of Multiphase Flow*, 126:103237, 2020.

[71] Can Selçuk, Arthur R Ghigo, Stéphane Popinet, and Anthony Wachs. A fictitious domain method with distributed lagrange multipliers on adaptive quad/octrees for the direct numerical simulation of particle-laden flows. *Journal of Computational Physics*, 430:109954, 2021.

[72] NC Shilton and K Niranjan. Fluidization and its applications to food processing. *Food structure*, 12(2):8, 1993.

[73] Petteri Sippola, Jari Kolehmainen, Ali Ozel, Xiaoyu Liu, Pentti Saarenrinne, and Sankaran Sundaresan. Experimental and numerical study of wall layer development in a tribocharged fluidized bed. *Journal of Fluid Mechanics*, 849:860–884, 2018.

[74] Kai Sotthewes, Han JGE Gardeniers, Gert Desmet, and Ignaas SM Jimidar. Triboelectric charging of particles, an ongoing matter: From the early onset of planet formation to assembling crystals. *ACS omega*, 7(46):41828–41839, 2022.

[75] Ken-ichiro Tanoue and Hiroaki Masuda. Electrostatic separation. In *Powder Technology Handbook, Fourth Edition*, pages 455–460. CRC Press, 2019.

[76] Martin Anton van der Hoef, M van Sint Annaland, NG Deen, and JAM Kuipers. Numerical simulation of dense gas-solid fluidized beds: a multiscale modeling strategy. *Annu. Rev. Fluid Mech.*, 40(1):47–70, 2008.

[77] J Antoon Van Hooft, Stéphane Popinet, Chiel C Van Heerwaarden, Steven JA Van der Linden, Stephan R De Roode, and Bas JH Van de Wiel. Towards adaptive grids for atmospheric boundary-layer simulations. *Boundary-layer meteorology*, 167:421–443, 2018.

[78] Henk Kaarle Versteeg. *An introduction to computational fluid dynamics the finite volume method, 2/E*. Pearson Education India, 2007.

[79] Stéphane Vincent, Jorge César Brändle De Motta, Arthur Sarthou, Jean-Luc Estivalezes, Olivier Simonin, and Eric Climent. A lagrangian vof tensorial penalty method for the dns of resolved particle-laden flows. *Journal of Computational Physics*, 256:582–614, 2014.

[80] Chunlei Wang, Guodong Liu, Zhanhu Zhai, Xinyao Guo, and Yao Wu. Cfd-dem study on the interaction between triboelectric charging and fluidization of particles in gas-solid fluidized beds. *Powder Technology*, 419:118340, 2023.

[81] Meurig W Williams. Triboelectric charging of insulating polymers–some new perspectives. *Aip Advances*, 2(1), 2012.

[82] Andrew M Wissink, Richard D Hornung, Scott R Kohn, Steve S Smith, and Noah Elliott. Large scale parallel structured amr calculations using the samrai framework. In *Proceedings of the 2001 ACM/IEEE conference on Supercomputing*, pages 6–6, 2001.

[83] Ying Wu, Daoyin Liu, Jinding Hu, Jiliang Ma, and Xiaoping Chen. Comparative study of two fluid model and dense discrete phase model for simulations of gas–solid hydrodynamics in circulating fluidized beds. *Particuology*, 55:108–117, 2021.

[84] Yishui Wu, GS Peter Castle, and Ion I Inculet. Induction charging of granular materials in an electric field. *IEEE transactions on industry applications*, 41(5):1350–1357, 2005.

[85] David P Young, Robin G Melvin, Michael B Bieterman, Forrester T Johnson, Satish S Samant, and John E Bussoletti. A locally refined rectangular grid finite element method: application to computational fluid dynamics and computational physics. *Journal of Computational Physics*, 92(1):1–66, 1991.

[86] Yongen Yu, Douglas H Rudd, Zhiling Lan, Nickolay Y Gnedin, Andrey Kravtsov, and Jingjin Wu. Improving parallel io performance of cell-based amr cosmology applications. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium*, pages 933–944. IEEE, 2012.

[87] Weiqun Zhang, Ann Almgren, Vince Beckner, John Bell, Johannes Blaschke, Cy Chan, Marcus Day, Brian Friesen, Kevin Gott, Daniel Graves, et al. Amrex: a framework for block-structured adaptive mesh refinement. *The Journal of Open Source Software*, 4(37):1370, 2019.