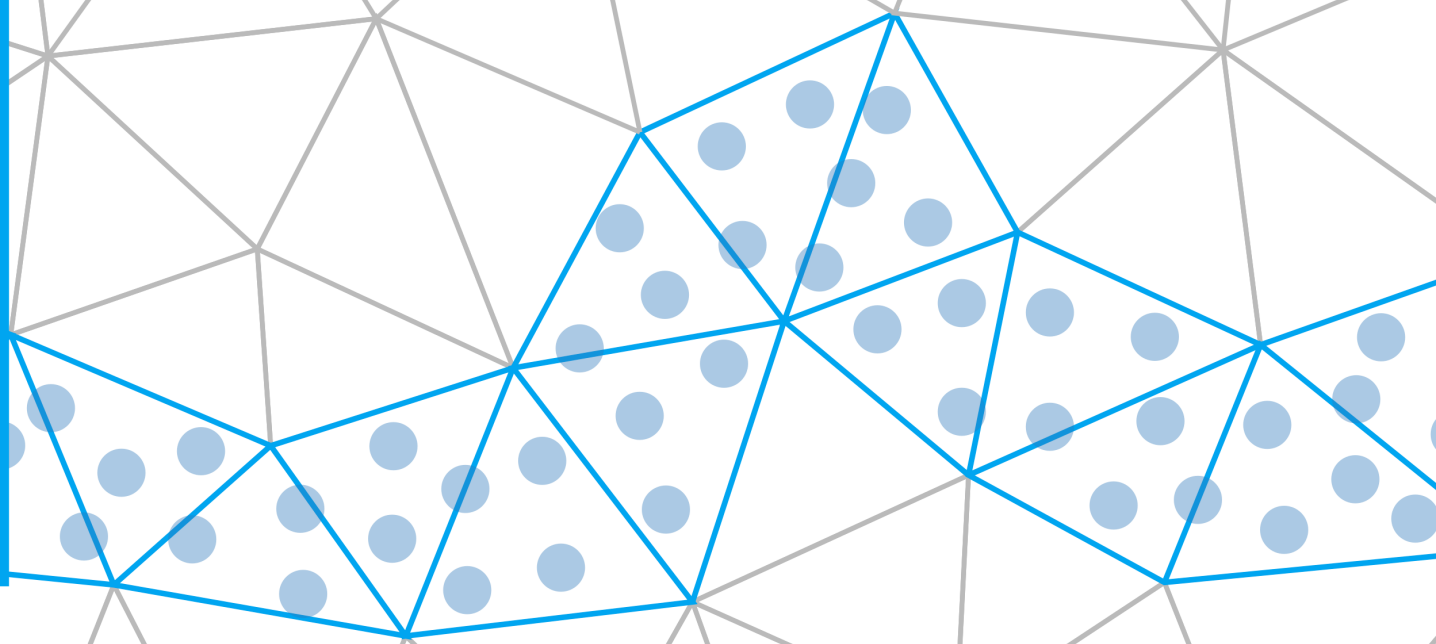
A hybrid particle-mesh method for simulating free surface flows

J.M. Maljaars

Technische Universiteit Delft



A hybrid particle-mesh method for simulating free surface flows

by

J. M. Maljaars

in partial fulfillment of the requirements for the degree of

Master of Science
in Civil Engineering

at the Delft University of Technology,
to be defended publicly on January 29, 2016.

Chair:	Prof. dr. ir. W. S. J. Uijttewaal
Thesis committee:	Dr. ir. R. J. Labeur
	Dr. M. Möller
	Ir. J. van den Bos

TU Delft Delft University of Technology

Abstract

Hybrid particle-mesh methods attempt to combine the advantages of Eulerian and Lagrangian methods: Lagrangian particles are used for the advection, whereas a Eulerian background grid is used for computing the particle interactions. Such a hybrid approach is expected to have several benefits when simulating flows involving free surfaces or material interfaces: the particles can be efficiently used to track the free surface, while the background grid can be used to solve the governing Navier-Stokes equations and impose the incompressibility constraint in a convenient manner. The prospects of these hybrid particle-mesh methods for simulating incompressible fluid flows involving a free surface are assessed in this thesis. More specifically, the feasibility of setting-up a numerical wave flume using hybrid particle-mesh methods is investigated.

Four main steps are common to all hybrid particle-mesh methods: particle-to-grid mapping, solving the equations at the background grid, grid-to-particle mapping and particle advection. Due to the large variety of possible model options, it is impossible to speak of *the* hybrid particle-mesh method. Main difference between the various approaches is how the particle-grid interaction is incorporated. Depending on the specific implementation, the hybrid particle-mesh methods can be either regarded to be a Eulerian method augmented with Lagrangian particles, or a Lagrangian method in which the Eulerian background grid only serves as a useful tool for solving the governing equations.
A specific implementation of the four main steps comprising the hybrid particle-mesh methods is presented, assigning relatively much importance to the background grid. The particle-to-grid mapping is presented in terms of a weighted least square mapping. In the specific implementation of this least square mapping, the grid nodes are considered to be sample points of the continuum. Following this strategy, an optimal quadrature rule can be used for numerical integration of the variational form arising from the finite element discretization of the governing equations at the background grid.
The specific implementation of advecting the particles in the grid velocity field, combined with the employed, admissible P_0P_1element, called for an additional mapping of the (divergence free) velocities to a (non-divergence free) continuous velocity field, a step which can at best be regarded to be an engineering solution. Assessing different advection schemes revealed the higher order scheme (RK3) to be an economical choice for doing the particle advection.

Various test cases were run, both for single-phase and two-phase problems, showing the advantages and the disadvantages of the developed model. The results obtained for the single-phase problems are in good agreement with analytical results or results obtained with established numerical methods. Employing an admissible element, pathological locking was effectively avoided in the method. Despite these good results, there is clearly some numerical diffusion present in the system. A more fundamental issue was encountered for the advection dominated lid driven cavity test, where the incompressibility constraint is clearly violated at particle level. As a result, unphysical gaps in the particle distribution are observed.
In general, the model results obtained for the two-phase benchmarks are again in good agreement with analytical or experimental results. With an additional mapping of the pressure gradient term, the sharp interface between materials (air-water) is well-maintained over time by the particles, although a residual particle oscillation was observed around the interface. Distinct advantage of the hybrid particle-mesh method is the implementation of kinematic boundary conditions. Since the mesh can be redefined every timestep without additional interpolation steps, the method can deal with large boundary displacements as shown for a solitary wave generated with a moving boundary. Finally, using the hybrid particle-mesh formulation, interfaces of complex topological shape are conveniently tracked by the particles. As such, the method was shown to be able so simulate a breaking wave on a submerged bar up to and including wave breaking.

Based on the thesis, it can be concluded that hybrid particle-mesh methods appear to be an attractive tool for simulating free surface flows and simulating the nearshore propagation of waves. Nevertheless, many

fundamental questions remain unanswered when considering hybrid methods. It is to be remarked that all these questions can be basically reduced to the question how to interpret the interaction between particles and grid. Future work should therefore primarily focus on this issue.

Preface

When following the course Ocean Waves, the lecturer briefly touched upon particle methods as a means of solving the continuum equations for fluid flows. Intrigued by the visually appealing results, what's in a picture, my interest in these methods was awakened. Choosing a research topic was therefore not that difficult when I got the opportunity to formulate a research project within the graduate programme on solid and fluid mechanics organized by the J.M. Burgerscentre (JMBC) for fluid mechanics and Engineering Mechanics (EM), the Dutch research school for solid mechanics. The freedom these two research schools offered in composing a master's programme track and the opportunity they offered for applying for a research grant was highly appreciated. This Master's Thesis on hybrid particle-mesh methods can be regared as the first fruit from this grant. Hopefully, many will follow in the upcoming years.

This product, result of approximately ten months of dedicated work, could not be established without the support and help of many persons. I am especially thankful to Prof. Uijttewaal who has supported me since the second year of my BSc. Motivating me to participate in the graduate programme offered by JMBC and EM and stimulating me to pursue a PhD, Prof. Uijttewaal often gave me the final push to overcome my hesitations. It is difficult to overestimate the valuable discussions with Robert Jan Labeur on the desired research direction and his patient explanation of the mathematical concepts.
I also would like to thank the other two members of the graduation committee, Matthias Möller and Jeroen van den Bos for their useful comments, either be it regarding the mathematics or the more practical side of the story. It was very useful to have a committee with members having different backgrounds.

Providing me with the health, the mental capabilities and an unconditionnally supportive family and friends, I would finally and foremost thank God for giving me everything what was needed, not only during the past year, but in fact during the 5 years period I am about to finish. Noting His hand in daily life and in the amazing complexity of nature, even a numerical study like this will lead to the humble confession: "What manner of Man is this, that even the wind and the sea obey Him?" (Mark 4:41)

Contents

1 Introduction

1.1 Background

Over the past century, the numerical modeling of flow problems in environmental fluid mechanics has made huge progress. Less than a century ago, it took Lorentz for example several months on mechanical calculators to compute the tidal flow in a network of one-dimensional channels representing the Dutch Waddenzee. A similar computation running at a modern desktop of average quality would take at most a few minutes nowadays. These advances in computational resources have opened the way to study and simulate fluid flows in more detail. In environmental fluid mechanics a clear tendency is therefore observed towards the development of models capable of simulating small scale problems. As an example of a small scale problem, one can think of the nearshore propagation of short waves, which is essentially the topic of this study. The nearshore wave propagation is not only of scientific interest because of the highly complicated process of wave breaking, but also of practical importance for the engineering community interested in the interaction between waves and coastal structures such as dunes, dikes and breakwaters. Over the past decades, a mainly empirical and/or experimental approach was used in engineering practice in order to study the interaction between waves and coastal structures as the physics-based, numerical simulation of those problems was computationally too demanding. Although such an approach has undoubtedly proven to work, it does give a poor insight in the underlying physics (empirical approach) or it is extremely expensive (experimental approach).

The potential of numerical models in this field of engineering is therefore well-understood: it can reduce the design costs of coastal structures, and avoid problems intrinsically related to physical model testing such as scaling problems. In an earlier conducted literature review [1], some numerical approaches were identified having the potential to simulate the nearshore wave propagation. Specific application in mind when assessing these methods was the interaction between waves and porous coastal structures such as breakwaters. This interaction can be regarded as a complicated problem, both from a physical and a numerical perspective as it involves many different scales (roughly from wave length to pore scale) and different physical phenomena, among others the water-air interaction, water-element interaction and element-element interaction. In the literature review it was concluded that the hybrid particle-mesh methods could have some benefits over fully Eulerian methods such as the volume-of-fluid (VOF) method [2] and fully Lagrangian methods such as smoothed particle hydrodynamics (SPH) [3] for the complicated application under consideration. As main advantages it was recognized that hybrid particle-mesh methods are in theory able to deal with multiple interacting materials (read fluid and solid elements) and strong deformations in a relatively cheap and efficient manner. However, some important pieces are missing in literature on hybrid particle-mesh methods or reported to be problematic. A first issue reported in for example the literature on the simulation of fluid flows with the Material Point Method (MPM), a popular hybrid particle-mesh method, is the pathological grid-locking effect. This phenomenon expresses itself in spurious pressure modes at the background grid, thus rendering the fluid flow computations extremely inaccurate (see e.g. [4] and [5]). Apart from this, it was recognized that there is a gap in knowledge on combining moving (wave) boundaries with hybrid methods. Accurate representation of such boundaries is however of utmost importance in order to set-up a numerical wave flume. Finally, in the literature review

on the relevant physical processes, air entrainment was found to play an important role in the energy dissipation of breaking waves. However, until now hybrid methods are generally applied for single-phase flows only. The only paper describing the simulation of two-phase air-water flow using hybrid particle-mesh methods stems from the graphics industry [6]. It is to be noted that in this industry the visual appearance of the solution is much more important than a proper representation of the physics. Therefore, it should be assessed whether hybrid particle-mesh methods are applicable for two- or multiphase problems.
So although promising at first sight, many open questions remain to be answered in order to assess whether hybrid particle-mesh methods are a feasible or, even better, a favourable way of simulating free surface flows.

1.2 Objective

Given the above considerations the applicability of hybrid particle-mesh methods for simulating free surface flows is investigated in this thesis. More specifically, as the long-term objective is to set-up an efficient numerical wave flume, this thesis attempts to assess whether hybrid particle-mesh methods are a suitable method to achieve this objective. The main question of this thesis is therefore formulated as:

What are the prospects of hybrid particle-mesh methods in simulating free surface flows and is it possible to set-up a numerical wave flume using a hybrid particle-mesh method?

The term 'prospects' should be interpreted as the (potential) advantages and disadvantages of hybrid particle-mesh methods over established Eulerian or Lagrangian methods.

1.3 Research questions

Three sub-questions are formulated in order to answer this question:

- Which particle-mesh methods are available and what are the key features of these methods?
- How to interpret the interaction between particles and mesh and how to implement a hybrid particle-mesh method?
- How do simulated results compare to analytical and/or experimental results? What are the advantages/disadvantages of the method?

By answering these research questions, an attempt is made to solve some issues related to hybrid particle-mesh methods as reported in literature (e.g. the grid-locking issue), furthermore it attempts to extend the range of applicability of the hybrid particle-mesh methods by implementation of a wave generating boundary and simulating two-phase air water flows. In order to achieve this, a hybrid particle-mesh method needs to be developed from scratch since there is no software available suiting the needs.

1.4 Methodology and outline

1.4.1. Methodology

Strategy followed in answering the research questions is to start with a broad view of the available numerical methods and narrowing the view to a specific implementation of a hybrid particle-mesh method. The first part of the thesis will therefore be aimed at judging the place of hybrid particle-mesh methods in the wealth of numerical tools available for the simulation of incompressible fluid flows. Special attention is paid to the difference between pure Lagrangian methods and the hybrid particle-mesh methods. Next step is then to compare the essentials of the different hybrid-particle mesh methods and to investigate the key steps of these methods. Based on this, the first implementation choices can be made, namely which hybrid particle-mesh implementation will be used.
Elaborating upon this choice, the scope is further narrowed by formulating the outline of a hybrid particle-mesh method. In this part, a generic formulation is sought for the identified key steps of the hybrid particle-mesh method. When formulating these steps, different fundamental model options are presented. Since the amount of model approaches in hybrid particle-mesh methods will turn out to be almost unlimited, a presentation of different fundamental model options can be beneficial in order to make some useful

recommendations regarding future research.
The next step is to present and discuss a specific implementation of the hybrid particle-mesh methods. Solutions to numerical intricacies inherently present in hybrid particle-mesh methods are presented and a detailed outline of the model implementation including the implementation choices is given.
The chosen implementation is assessed by running different benchmark tests. These tests are chosen such that 1) they reveal the strengths and weaknesses of the hybrid particle-mesh method and/or 2) the results can be compared to analytical results or results obtained from literature. In this way, the performance of the developed method can be compared to other methods such as the fully Lagrangian SPH method. The first benchmark tests will deal with single-phase flows. Although mainly of academic interest, these benchmarks will give insight in the fundamental properties of the developed model. Finally, benchmark tests are run for two-phase flows. These benchmark tests are indeed more targeted at answering the question whether hybrid particle-mesh methods are a suitable tool for setting-up a numerical wave flume. Therefore, when dealing with those benchmark tests the implementation of wave generating boundaries is discussed, as well as the inclusion of open boundaries.

1.4.2. Outline

The thesis is organized following the methodology presented above. An overview of the hybrid particle-mesh methods is given in Chapter 2. In Chapter 3 a generic framework for a hybrid particle-mesh model formulation is presented. Special attention is paid to the different available model approaches and the interaction between particles and grid is put into perspective. Chapter 4 describes the implementation of a hybrid particle-mesh method, discussing both theoretical and practical issues. In Chapter 5 the model is tested for some academic single-phase fluid flow test cases. In Chapter 6 the step is made towards the simulation of real life problems, by considering a few test cases for free surface/two-phase flows. In this chapter, the implementation of a wave boundary is presented and compared to results for a solitary wave. As a final example, wave breaking over a submerged bar is considered. In Chapter 7 the main research question is answered and an outlook for future research is given.

2

Hybrid particle-mesh methods

In this chapter, some hybrid particle-mesh methods used in the computational fluid dynamics (CFD) world are introduced. However, before discussing these methods, it has to be clear which equations are attempted to be solved by these methods. Therefore, the chapter starts by introducing the governing equations for a Newtonian and incompressible fluid. After that, hybrid particle-mesh methods are briefly compared with the full particle methods, of which SPH is probably known best in the CFD world. In Section 2.3, three different particle-mesh methods are introduced. Apart from discussing their key features, some issues specific for these methods are discussed as well.

2.1 Numerical modeling of fluid flows: the essentials

2.1.1. Continuum equations

Before focusing on hybrid particle-mesh methods it is important to know what such a method actually does, or to put it more generic: what does numerical modeling of fluid flows essentially mean? At the basis of all continuum models are the conservation laws: conservation of mass, conservation of momentum and conservation of energy. When formulating these equations, it is important to know in which frame of reference the equations are formulated. Two different approaches are distinguished. In the first approach, the Eulerian approach, a control volume[1] is considered which is fixed both in time and space. The equations describe the changes of the material properties in the control volume over time. The equations in a Eulerian frame of reference therefore describe the mass, momentum and energy inflow and outflow of the control volume as well as the optional sources and sinks for these quantities inside the control volume. In the second approach, the Lagrangian approach, a material volume [2] itself is followed over time. In this approach the change of material volume properties over time is considered, where it is emphasized that the material volumes are impermeable so that they do not overlap. Fundamental difference between the Eulerian and Lagrangian approach is thus the frame of reference which is used: in the Eulerian approach a fixed frame of reference is employed, whereas the frame of reference itself is moving with the material volumes in the Lagrangian approach. This key difference between the two approaches is sketched in Figure 2.1.

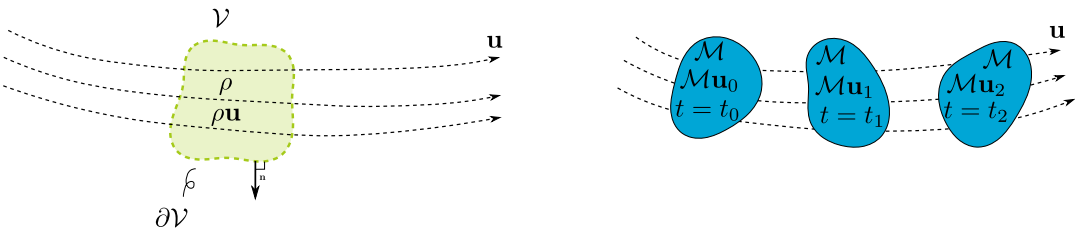


Figure 2.1: Principle sketch of a Eulerian (left) and a Lagrangian (right) frame of reference.

[1] This typically leads to a grid representation of the continuum in mesh-based methods.
[2] For particle methods and hybrid particle-mesh methods, these material volumes are represented by a set of particles.

For the sake of simplicity, the relevant equations are derived in a Eulerian frame of reference. From these, the equations in a Lagrangian frame of reference are easily derived. To this end, consider the Eulerian control volume $\mathcal{V} \subset \mathbb{R}^d$ where $\mathbb{R}^d$ denotes the domain of interest having spatial dimension d. The boundary of the control volume $\mathcal{V}$ is denoted as $\partial\mathcal{V}$ having the outward pointing normal $\mathbf{n}$, Figure 2.1. Furthermore, define $I = (t_0, t_1)$ to be the considered time interval. Ignoring any sources and sinks in the volume $\mathcal{V}$, it can be stated that the total mass change equals the net inward transport of mass through the boundary during the considered time interval I, and so:

$$\int_{\mathcal{V}} \Delta\rho \mathrm{d}V = -\int_I \oint_{\partial V} \rho\mathbf{u} \cdot \mathbf{n} \mathrm{d}\Gamma \mathrm{d}t \tag{2.1}$$

in which $\Delta\rho$ is the density increment in the time interval I, $\mathbf{u}$ denotes the velocity, and $\mathbf{n}$ is the outward pointing normal vector at the boundary $\partial\mathcal{V}$.
Applying the divergence theorem to the right hand side results in:

$$\int_{\mathcal{V}} \Delta\rho \mathrm{d}\Omega = -\int_I \int_{\mathcal{V}} \nabla \cdot (\rho\mathbf{u}) \mathrm{d}\Gamma \mathrm{d}t \tag{2.2}$$

Assuming $\mathbf{u}$ and ρ to be differentiable and noting that this equality must hold for arbitrary volumes $\mathcal{V}$ results in the continuity equation in conservative form:

$$\frac{\partial\rho}{\partial t} + \nabla \cdot (\rho\mathbf{u}) = 0 \tag{2.3}$$

which becomes after rewriting the second term:

$$\frac{\partial\rho}{\partial t} + \mathbf{u} \cdot \nabla\rho + \rho\nabla \cdot \mathbf{u} = 0 \tag{2.4}$$

$$\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho\nabla \cdot \mathbf{u} = 0 \tag{2.5}$$

The term $\frac{\mathrm{D}\rho}{\mathrm{D}t}$ is recognized as the Lagrangian derivative of the density. Indeed, this term describes the density variation of a *moving* particle caused by pressure, temperature or salinity changes. However, these changes can usually be ignored for environmental fluid mechanics applications since the density variations are in general negligible compared to the reference density. Under this approximation, the continuity equation reduces to:

$$\nabla \cdot \mathbf{u} = 0 \tag{2.6}$$

In a similar vein, the momentum conservation equation can be derived. The change of momentum in a control volume is not only governed by the net inflow of momentum, but also by the forces acting on the volume $\mathcal{V}$ and its surface $\partial\mathcal{V}$. Thus the momentum balance for the control volume becomes:

$$\int_{\mathcal{V}} \Delta(\rho\mathbf{u}) \mathrm{d}V = \int_I \int_{\mathcal{V}} \mathbf{f} \mathrm{d}V \mathrm{d}t - \int_I \oint_{\partial\mathcal{V}} (\rho\mathbf{u} \otimes \mathbf{u}) \cdot \mathbf{n} \mathrm{d}\Gamma \mathrm{d}t - \int_I \oint_{\partial\mathcal{V}} \boldsymbol{\sigma}\mathbf{n} \mathrm{d}\Gamma \mathrm{d}t \tag{2.7}$$

In this equation, the advective part is given by the dyadic product $\rho\mathbf{u}\otimes\mathbf{u} = \rho u_i u_j$. Furthermore, $\mathbf{f}$ represents the body forces acting on the fluid volume $\mathcal{V}$ (e.g. gravity) and $\boldsymbol{\sigma}$ is the Cauchy stress tensor acting on the control volume surface $\partial\mathcal{V}$. This stress tensor is a result of the pressure forces and the viscous shear forces and for an incompressible, Newtonian fluid defined as:

$$\boldsymbol{\sigma} = -p\mathbf{I} + 2\mu\nabla^s\mathbf{u} \tag{2.8}$$

with p the pressure, $\mathbf{I}$ the identity matrix, μ the dynamic viscosity and ∇^s is the symmetric gradient operator $\nabla^s(\cdot) = \frac{1}{2}\nabla(\cdot) + \frac{1}{2}\nabla(\cdot)^T$. Applying the divergence theorem to the surface integrals, and noting that the equality must hold for arbitrary volumes $\mathcal{V}$, the momentum equations in conservative form are obtained:

$$\frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\mathbf{u} \otimes \mathbf{u}) = \mathbf{f} - \nabla p + \nabla \cdot (2\mu\nabla^s\mathbf{u}) \tag{2.9}$$

Using the identity $(\mathbf{a} \otimes \mathbf{b}) \cdot \mathbf{c} = \mathbf{a} \otimes (\mathbf{b} \cdot \mathbf{c})$, the left hand side can be expanded as:

$$\begin{aligned} \frac{\partial(\rho\mathbf{u})}{\partial t} + \nabla \cdot (\rho\mathbf{u} \otimes \mathbf{u}) &= \rho\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u}\frac{\partial \rho}{\partial t} + \rho\mathbf{u}\nabla \cdot \mathbf{u} + \rho\mathbf{u} \cdot \nabla\mathbf{u} + (\mathbf{u} \otimes \mathbf{u}) \cdot \nabla\rho \\ &= \rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\nabla \cdot \mathbf{u} + \mathbf{u}\left(\frac{\partial \rho}{\partial t} + \rho\nabla \cdot \mathbf{u} + \mathbf{u} \cdot \nabla\rho\right) \\ &= \rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u}\nabla \cdot \mathbf{u} + \mathbf{u}\left(\frac{\mathrm{D}\rho}{\mathrm{D}t} + \rho\nabla \cdot \mathbf{u}\right) \end{aligned} \tag{2.10}$$

Combined with the continuity equation in generic form, Eq. (2.5), the momentum equations can be rewritten as:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u} \cdot \nabla\mathbf{u} = \mathbf{f} - \nabla p + \nabla \cdot (2\mu\nabla^s\mathbf{u}) \tag{2.11}$$

Furthermore, the first two terms at the left hand side can be combined to:

$$\rho\frac{\partial \mathbf{u}}{\partial t} + \rho\mathbf{u} \cdot \nabla\mathbf{u} \equiv \rho\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} \tag{2.12}$$

And so the momentum equations for an incompressible, Newtonian fluid in Lagrangian form become:

$$\rho\frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = \mathbf{f} - \nabla p + \nabla \cdot (2\mu\nabla^s\mathbf{u}) \tag{2.13}$$

Eq. (2.6) forms together with Eq. (2.13) the Navier-Stokes equations for incompressible, Newtonian fluids.
An important difference between the momentum equations in Eulerian form and those in Lagrangian form is the way in which the advection term is expressed. In a Eulerian frame of reference, this results in a non-linear term, whereas advection in Lagrangian form is incorporated in the material derivative, thus avoiding the non-linear term.

2.1.2. Boundary and initial conditions

Together with the continuity equation, Eq. (2.6), the momentum equations, Eq. (2.13) form the starting point for simulating incompressible flows. However, in order to solve these equations, proper boundary and initial conditions have to be specified at the boundary of the domain. For the Navier-Stokes equations, boundary conditions have to be specified in both the normal and the tangential direction at the boundary $\partial\Omega$ for all $t \in I$, where $\partial\Omega$ denotes the boundary of the domain Ω. Practically, this means that either the velocity in normal and (or) tangential direction is specified (the Dirichlet BC's) *or* the stress in respectively the normal and (or) tangential direction needs to be provided (since the stress tensor scales with the gradient of the velocity, this becomes a Neumann BC). Applying a Dirichlet BC and a Neumann BC simultaneously in either the normal or tangential direction will lead to ill-posed problems.
Probably the best-known boundary condition for the Navier-Stokes equations is the Dirichlet BC in both the normal *and* the tangential direction, i.e.:

$$\mathbf{u} = \mathbf{u}_D \qquad \text{on} \quad \Gamma_D \times I \tag{2.14}$$

where Γ_D denotes the part of the boundary where the Dirichlet BC is applied, and $\mathbf{u}_D$ the velocity at the boundary. Setting this value to zero, results in the no-normal flow and no-slip conditions, thus representing a fixed, closed wall.
Instead of specifying the tangential or normal velocity component, the respective component of the traction vector can be defined. Open boundaries are for example specified by setting $\boldsymbol{\sigma}\mathbf{n} = \mathbf{0}$.
For the Euler equations, in which the viscous term is ommitted and hence shear stresses in the fluid are absent, the problem is already well-posed by only imposing a boundary condition in the normal direction. The Dirichlet BC thus becomes:

$$\mathbf{u} \cdot \mathbf{n} = \mathbf{u}_D \cdot \mathbf{n} \qquad \text{on} \quad \Gamma_D \times I \tag{2.15}$$

This condition prevents penetration of the flow through the wall, arbitrary slip is however permitted.
The complexity of open boundaries also reduces for the Euler equations. Such boundaries are specified by a zero pressure at the corresponding part of the boundary, that is:

$$p = 0 \qquad \text{on} \quad \Gamma_O \times I \tag{2.16}$$

If the problem only contains velocity boundaries, the pressure is solved up to a constant. The reason for this is that only the pressure *gradient* appears in the equations, thus, the exact value of the pressure is not important but only the pressure gradients do matter. In order to obtain a single valued pressure field, the pressure is often prescribed in one point of the domain.
Furthermore, it is noted that $\Gamma_P \cup \Gamma_D \cup \Gamma_O = \partial\Omega$ and $\Gamma_P \cap \Gamma_D \cap \Gamma_O = \emptyset$, so the union of the boundary parts should cover the whole boundary $\partial\Omega$ whereas the intersection of the boundary parts should be empty, Figure 2.2.

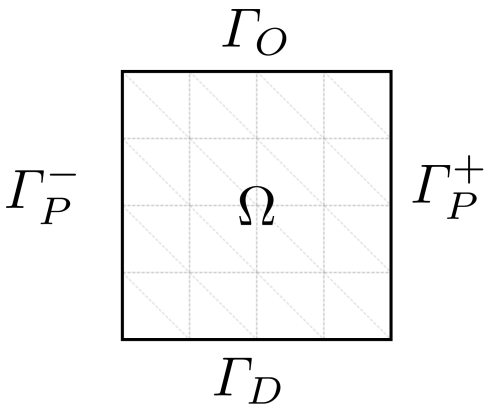


Figure 2.2: Schematic overview of domain definitions.

Apart from the boundary conditions, an initial condition for the velocity field should be imposed in the domain Ω. For incompressible fluids, the initial velocity field has to be divergence free. In other words, the initial condition should satisfy Eq. (2.6). The initial condtion can thus be written as:

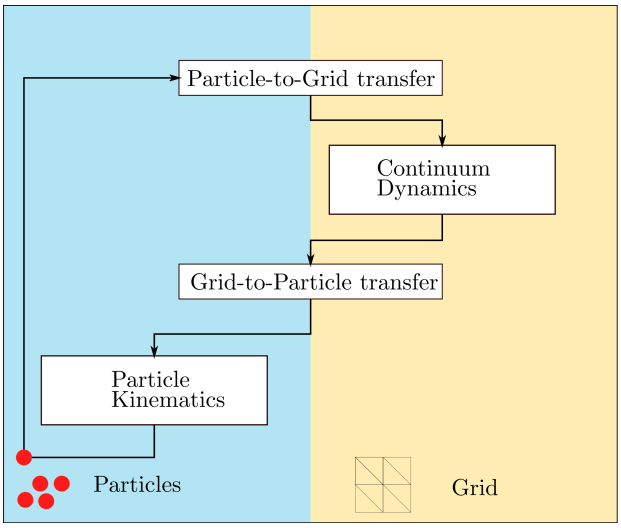
$$\begin{aligned} \mathbf{u}(\mathbf{x}, t_0) &= \mathbf{u}_0 \qquad \text{in} \quad \Omega, \quad \text{such that} \\ \nabla \cdot \mathbf{u}_0 &= 0 \end{aligned} \tag{2.17}$$

with $\mathbf{u}_0(\mathbf{x})$ the initial velocity field.
Although the pressure is time-dependent, no initial condition needs to be specified for this quantity. This can be explained by the fact that no time derivative of the pressure appears in the governing equation [7], but it is probably better understood by realizing that the pressure is not related to any constitutive equation but merely acts as a Lagrange multiplier to enforce the incompressibility constraint for incompressible flows [8].

2.2 Lagrangian versus hybrid Lagrangian/Eulerian methods

In the field of modeling of (violent) free surface flows, Lagrangian methods such as SPH have become increasingly popular. This can be attributed to several attractive features of the Lagrangian methods. As explained in the previous section, material volumes are followed over time in Lagrangian methods, these material volumes are usually represented as particles. Using such a particle representation of the continuum has the distinct advantage that moving material boundaries are easily tracked. Indeed, the material interface is easily deduced from the configuration of the material volumes so that Lagrangian methods are ideally suitable for simulating problems having interfaces of complex topology. As typical applications for Lagrangian methods one can therefore think of impact problems (see amongst many others e.g. [9], [10]) and free surface flows (see amongst many others e.g. [3], [11]). The material volume representation of the continuum makes an extension of such methods to multi-material problems also relatively straightforward. In [11] and [12] the Lagrangian SPH method was applied to two-phase fluid simulations with

large density differences. A final advantage of Lagrangian methods to be mentioned is the straightforward implementation of the advective term [13]: this term is simply modeled as a particle motion instead of a flux through element boundaries in the Eulerian frame of reference.
Despite the advantages of fully Lagrangian methods over Eulerian methods, there are some clear disadvantages innate to Lagrangian formulations. Major disadvantages of the Lagrangian methods are the implementation of boundary conditions (one of the 'grand-challenges' as defined by the SPHeric community [3]) and the computational expenses. The latter is mainly caused by the expensive neighbor-searching algorithm required for computing gradients [10]. Specific to applications for incompressible fluid flow problems, it should be mentioned that enforcing the incompressibility constraint is far from being trivial in pure Lagrangian methods. Therefore, it is common practice to assume the fluid to be weakly compressible, although in relatively recent SPH literature attempts have been made to assume the fluid to be strictly incompressible (see e.g. [14]).
Hybrid Lagrangian/Eulerian methods (in this thesis they are coined 'hybrid particle-mesh methods') attempt to combine the advantages of Eulerian and pure Lagrangian methods, while eliminating their drawbacks as much as possible. In short, this means that hybrid methods attempt to offer the 'flexibility of SPH with the efficiency of a Eulerian approach' [15]. Although there is a vast range of hybrid particle-mesh methods (a few main categories will be discussed shortly), they all have in common that discrete particles are interacting with a background grid, Figure 2.3.



$$(\mathbf{x}_p, \mathbf{u}_p) \rightarrow \rho, \mathbf{u}$$

$$\begin{cases} \rho \dfrac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = \mathbf{f} - \nabla p + \nabla \cdot (2\mu \nabla^s \mathbf{u}) \\ \nabla \cdot \mathbf{u} \\ BC's \end{cases}$$

$$\mathbf{u} \rightarrow \mathbf{u}_p$$

$$\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t} = \mathbf{u}$$

Figure 2.3: Main steps in hybrid particle-mesh methods (left panel, modified from [16]) and abstract mathematical formulation of hybrid particle-mesh method for incompressible flows (right panel). Variables without subscript denote grid variables, variables with subscript p denote particle variables.

According to this figure, a timestep in a hybrid method comprises four main steps: transferring data from particles-to-grid in order to construct a density and a velocity field, solving the dynamic equations at the background grid, map the solution back to the particles in order to update the particle velocity and finally do the particle kinematics by advecting the particles. In Chapter 3 these steps are more extensively discussed. Main advantage of the hybrid approach is that the different steps in a numerical algorithm can be done in the most appropriate representation. Indeed, advection (the kinematic part) is most easily done at particle level so in a Lagrangian manner, whereas the dynamic part of the algorithm (so for CFD applications this means solving the governing Euler or Navier-Stokes equations) is most easily done at grid level, making use of optimized and fast grid-based solvers [15]. In fact, any well-known discretization technique can be used for this grid step (e.g. finite difference, finite volume, finite elements). In doing so, the background grid is used as a means to solve for the interaction between particles. Using a background grid has two additional advantages. The first advantage is that the incompressibility constraint can be enforced as in regular grid-based methods (see e.g. [6], [15], [17] and [18]). This means that it is relatively straightforward to obtain a divergence free velocity field at the *grid level* (i.e. satisfying Eq. (2.6)). A

[3] https://wiki.manchester.ac.uk/spheric/index.php/Main_Page

second advantage of hybrid particle-mesh methods is that the boundary conditions can be conveniently enforced at the background grid [19].
Compared to fully Lagrangian methods, the hybrid particle-mesh methods are computationally less demanding (see e.g. [10], [20], [15]). Main reason for this is that an expensive neighbor-particle searching algorithm is avoided (since the particle-particle interaction is solved for at the background grid) and the size of the resulting matrices to solve scale with the mesh resolution instead of the particle spacing.
In short, it can be said that hybrid particle-mesh methods have some distinct advantages over the fully Lagrangian methods. The main strength of this type of methods is that it allows to perform the algorithmic steps in the most convenient representation in an efficient way, although it can be anticipated that transfering information between representations is an issue itself. As such, hybrid particle-mesh methods appear to be a promising tool for simulating free surface flows. Nevertheless, it will become clear at the end of this chapter and in the remainder of the report that these hybrid methods have their own limitations and drawbacks.

2.3 Overview of hybrid particle-mesh methods

2.3.1. Particle-in-Cell Method

The development of hybrid particle-mesh methods traces back to the 1950s when Harlow and coworkers developed the Particle-in-Cell (PIC) methods at Los Alamos National Laboratory ([21], [22]). Motivation behind the development of this method was to obtain a method able of combining the advantages of a Eulerian and a Lagrangian approach [23]. Whereas a Lagrangian (grid-based) approach is perfectly suitable for tracking fluid interfaces, the mesh deformation quickly becomes problematic for highly distorted flows. Although the Eulerian methods, having a fixed mesh, can deal with large distortions, these methods fall short in maintaining a sharp interface between different materials as they tend to diffuse the interface.
As an answer to the controversy between these two computational viewpoints, the PIC method was developed by sampling the continuum as a set of particles on a Eulerian background grid. A mass, velocity and a position is assigned to the particles. As outlined in the previous section, each computational cycle in PIC starts with the construction of a density and a velocity field on the background grid, given the particle distribution [21]:

$$\rho_i = \frac{1}{V_i} \sum_{p \in P_{N_i}} N_i(\mathbf{x}_p) m_p \tag{2.18}$$

$$\mathbf{u}_i = \frac{1}{\rho_i V_i} \sum_{p \in P_{N_i}} N_i(\mathbf{x}_p) m_p \mathbf{u}_p \tag{2.19}$$

In these equations the particle related quantities are denoted with a subscript p, so the particle mass is denoted as m_p and the particle velocity as $\mathbf{u}_p$. Furthermore, the subscript i denotes the locations of the nodal degrees of freedom at the background grid, so that V_i is interpreted as the volume associated to node i and N_i denotes the interpolation function or the so-called 'assignment function' [24]. The summation runs over the set of particles in the support domain of the interpolation function, that is:

$$P_{N_i} = \{p \ : \ N_i(\mathbf{x}_p) \neq 0\} \tag{2.20}$$

Common choices for the assignment function are the nearest grid point (NGP) interpolation, or the bilinear interpolation.
When the assignment function satisfies partition of unity:

$$\sum_i N_i(\mathbf{x}_p) = 1 \tag{2.21}$$

the mapping conserves mass and linear momentum, i.e. the grid mass and momentuem are equal to the particle mass and momentum. This is easily seen when summing over all grid unknowns i, yielding for the

mass conservation:

$$\sum_i \rho_i V_i = \sum_i \sum_{p \in P_{N_i}} N_i(\mathbf{x}_p) m_p \tag{2.22}$$

$$\sum_i m_i = \sum_{p \in P_\Omega} m_p \tag{2.23}$$

$$\tag{2.24}$$

and for the linear momentum:

$$\sum_i \mathbf{u}_i \rho_i V_i = \sum_i \sum_{p \in P_{N_i}} N_i(\mathbf{x}_p) \mathbf{u}_p m_p \tag{2.25}$$

$$\sum_i \mathbf{u}_i m_i = \sum_{p \in P_\Omega} \mathbf{u}_p m_p \tag{2.26}$$

$$\tag{2.27}$$

where P_Ω is defined as:

$$P_\Omega = \{p \,:\, \mathbf{x}_p \in \Omega\} \tag{2.28}$$

After constructing the velocity and the density field at the background grid, the next step is to solve the governing equations in Lagrangian form at the background grid. These equations are usually the mass and momentum equations for compressible media, since the PIC method was originally developed for simulating fluid applications with significant compression (for Mach > 0.3), thus allowing explicit integration of the equations ([23],[25]). Although any of the well-established grid discretization techniques can be used for this step, the finite difference approach is, probably for historic reasons, usually employed.
After solving the governing equations at the background grid for the velocity, particle properties are updated. Since this thesis deals with incompressible flows, the particle mass is constant and hence the only property to be updated is the particle velocity. This is done as:

$$\mathbf{u}_p = \sum_i N_i(\mathbf{x}_p) \mathbf{u}_i \tag{2.29}$$

with $\mathbf{u}_i$ the (updated) grid velocity.
Given this velocity interpolation, the particle position can be updated. In its simplest form, this can be done as:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \mathbf{u}_p^{n+1} \Delta t \tag{2.30}$$

where $\Delta t = t^{n+1} - t^n$ denotes the discrete time step, with the superscripts $n+1$ and n respectively denoting the new and the old time level.
Due to the repeated back-and-forth mapping between particles and grid, the PIC scheme is known to be very diffusive, an issue already noted by the developers [21]. As a solution to this (extremely) diffusive behavior, the fluid implicit particle (FLIP) method was derived as an alternative to the PIC method. Before discussing the FLIP method, a few important remarks regarding PIC have to be made.
Firstly, particles are merely used as a means to advect physical quantities from one cell to another. Indeed, particles are only assigned a mass, all other physical quantities at particle level are every time step entirely dictated by the grid solution. It can thus be questioned whether the particles are really a representation of the continuum, or just a 'trick' to deal with 1) advection in a convenient way and 2) keeping sharp interfaces for highly distorted flows. The latter feature of PIC was in fact exploited to come up with the marker-and-cell method (MAC) method, which was also developed at Los Alamos and designed to simulate incompressible flows [23] . In the MAC method, passive particles are added, i.e. particles interact with the grid, but there is no feedback from particles to the grid.
Secondly, although the PIC method was labeled to be 'mainly of historical interest' [24] and 'obsolete' [26] after the development of FLIP, Jiang *et al.* [16] shed new light on the interaction between particles and grid. Their reasoning is that although linear momentum is conserved in the PIC method, angular momentum is not conserved in the mapping from grid to particles, thus rotational motion is severely damped. Key idea

in mitigating this problem is to consider the particles to be rigid bodies and to augment both the particle to grid transfer as well as the grid to particle transfer with a term to take the angular momentum into account in order to preserve this quantity. Their method was termed 'affine particle-in-cell' (APIC). As the research was conducted within the graphics industry, the method was only visually inspected. However, since the method has a firm theoretical basis, the reinvention of the traditional PIC method can be useful for engineering practice as well.

2.3.2. Fluid Implicit Particle Method

In general, the fluid implicit particle method (FLIP) is regarded to be the logical successor of the PIC method [27]. Since its development in the 1980s, the FLIP method has been applied for a vast range of problems. To name a few: the FLIP method has been used for the simulation of sand dynamics [17], the interaction between fluid and obstacles [28], the simulation of two-phase flows [6] and the physics-based simulation of snow [29]. As pointed out by several authors, the success of FLIP can be attributed to the fact that the method combines an accurate, non-diffusive transport of flow quantities together with an easy treatment of the free surface boundary and the incompressibility constraint ([17], [18]). Based on the above references, it can be concluded that especially the graphics industry has widely adopted the FLIP method for physics based fluid animations. Only very recently, a paper was published in which FLIP (in this paper termed 'full PIC') is used as an efficient tool for simulating the force of waves on rigid bodies [15].

At first sight, FLIP is only a slight variation to the PIC method. The method thus roughly follows the steps described in the previous section: mapping the state variables (density and velocity) to the background grid, solve the background grid equations and updating the particle properties. However, instead of *overwriting* the particle velocity with Eq. (2.29) every time step, the particle velocity is *updated* as [30]:

$$\mathbf{u}_p^{n+1} = \mathbf{u}_p^n + \sum_i N_i(\mathbf{x}_p)\left(\mathbf{u}_i^{n+1} - \mathbf{u}_i^n\right) \tag{2.31}$$

in which $N_i(\mathbf{x})$ is the assignment function corresponding to grid node i and $\mathbf{u}_p$ and $\mathbf{u}_i$ are particle and grid velocities respectively.

Indeed, where in the classical PIC method only the mass is carried by the particles between consecutive time steps, in FLIP this also holds true for the particle velocity. Each time step this particle velocity is *updated* with an increment instead of *overwritten* by the grid velocity. In short, all of the fluid properties are assigned to the particles and so FLIP can be regarded as a fully Lagrangian description of the fluid. Realizing this, it becomes clear that the seemingly minor difference between PIC and FLIP is rather a fundamental difference, namely that of considering particles only as a simple means to advect particle properties (PIC) or as a representation of the fluid (FLIP). In other words, it can be concluded that the particles are assigned much more importance in FLIP compared to PIC. Hence, the role of the grid as a computational convenience for computing the particle interactions is further employed in FLIP [30]. Because all information is carried by the Lagrangian particles and no information is stored on the grid between consecutive time steps, the grid can be conveniently redefined every time step. It can be anticipated that this feature can be extremely helpful for setting up a numerical wave tank, using moving boundaries.

Although the FLIP method efficiently avoids the numerical diffusion present in the classical PIC approach, it certainly has another main disadvantage: numerical noise can develop at particle level, rendering the computation inaccurate or even instable ([16], [27], [31],). This issue, in fact present in all hybrid particle-mesh methods, but especially pronounced in FLIP will be more extensively discussed at the end of this chapter when investigating the issues encountered in hybrid particle-mesh approaches.

When applied to free surface flows, FLIP is often used in combination with a level set approach to locate the free surface. Examples of this can be found in e.g. [6], [17], [18], [32]. Idea of a level set is to construct a scalar field, usually termed a 'signed distance function', being negative in the air region and positive in the water region (or vice versa) and thus the free surface is located at the zero isocontour of the signed distance function. As shown in aforementioned references, particles can be conveniently used to construct such a function. A simple approach inspired by the work in [6] is explained in Appendix C. On the free surface, usually the dynamic boundary condition $p = 0$ is enforced, although the surface tension can be included as well. The kinematic free surface boundary is trivially satisfied in a Lagrangian setting.

Level set functions for detecting the free surface are not restricted to hybrid particle-mesh methods, but are

also extensively used in purely Eulerian methods such as VOF. However, whereas an additional advection equation has to be solved in a Eulerian setting in order to move the level set, in a hybrid particle-mesh method the level set is trivially 'advected' by the moving particles [6]. This also avoids some difficulties inherently present in the Eulerian level set formulation such as a lack of volume conservation when advecting the level set. Thus it is concluded that particles form a useful means for tracking free surfaces and/or interfaces.

2.3.3. Material Point Method

The material point method (MPM) is an extension to the FLIP method and was originally developed to simulate history dependent materials such as soils or granular material [33]. Whereas in a Newtonian fluid the constitutive equation (Eq. (2.8)) is history independent, for granular materials the constitutive equation may depend on e.g. the plastic strain [33]. A hybrid Lagrangian-Eulerian method is a perfect tool for simulating such materials since the particles (termed material points) can carry those history-dependent variables (such as strains and stresses) and can efficiently deal with large deformations of granular bodies. In literature the numerical modeling of granular and geotechnical materials using the material point method got extensive attention (see e.g. [4], [19], [34], [35]). Apart from this, the material point, has also been applied to simulate high energy impact problems ([10], [12]), fluid-membrane interaction ([5], [36]), and it has been used as a tool to couple atomistic models to continuum models [37] to name only a few applications. Of special interest for this thesis is the application of MPM to fluid mechanics problems, for which a few references are available ([4], [5], [20]). Discussion of these papers will be extended to the end of this section.

The MPM procedure roughly follows the same path as the procedure for the PIC and the FLIP method. There are however a few important differences. The first difference to be mentioned is due to the spatial discretization of the governing equations: in MPM the weak form of the equations is derived (for details on deriving the weak form, the reader is referred to Chapter 3), the integrals resulting from this procedure are approximated using the particle properties (see Section 3.1 for an example). So instead of first constructing the background grid field for the density and the velocity and subsequently solving the governing equations at the background grid, in MPM the particle-to-grid mapping is actually performed when solving the governing equations. This also means that gradients are not evaluated at the background grid as done in PIC and FLIP, but the shape functions are first differentiated, and subsequently mapped to the background grid given the value at the particle location [38]. As an example, consider the computation of the nodal internal force in MPM from the particle stresses [39]:

$$\mathbf{F}_i^{\text{int}} = -\sum_p \nabla N_i(\mathbf{x}_p) V_p \boldsymbol{\sigma}_p \tag{2.32}$$

where $\mathbf{F}_i^{\text{int}}$ the internal force vector at node i and $\boldsymbol{\sigma}_p$ the stress tensor of particle p and V_p the particle volume which acts as the integration point weight.

It is clear that the solution strategy used in MPM emphasizes the particle character of the continuum even more than PIC or FLIP.

Although MPM evolved from PIC and FLIP, the fluid mechanics roots seems interestingly enough to be a little bit forgotten when applied to nearly incompressible (free surface) flows as done by e.g. Mast *et al.* [4], Hamad [5] and Mao *et al.* [20]. All these authors assume the fluid to be weakly compressible, that is, they assume the pressure to be related to the density by an equation of state of the form:

$$p = B\left[\left(\frac{\rho}{\rho_0}\right)^{\gamma} - 1\right] \tag{2.33}$$

with p the pressure and B is often computed based on an artificial speed of sound [3], furthermore, ρ_0 is the reference density and γ is a constant and usually taken equal to 7.

More importantly, in all papers an equal-order interpolation is used for the pressure and the velocity unknowns, i.e. the same shape functions are used for both the velocity and the pressure. These choices have two important consequences for the modeling of incompressible flows:

- For weakly-compressible material the Courant-Friedrichs-Lewy (CFL) condition for explicit time step-

ping schemes is governed by the propagation speed of sound through the medium, so:

$$\frac{c\Delta t}{h} \leq \epsilon \qquad \text{where } c = \sqrt{\frac{K}{\rho}} \tag{2.34}$$

with K the bulk modulus of the material, ρ the material density, h a characteristic length scale for the grid resolution and Δt the model time step.
This condition leads to restrictively small time steps as the speed of sound is much higher than the flow velocity in environmental fluid mechanics problems. Timestep in weakly-incompressible models is therefore typically in the order of $1E-5$ s.

- Furthermore, a pathological locking is observed in the model results, leading to unphysical pressure fields. This is in fact the result of using (unmodified) equal-order interpolations. In Chapter 3 this issue will be further examined.

The first issue is usually mitigated by assuming an artificial compressibility being several orders of magnitude lower than the true bulk modulus of water. The second issue is usually mitigated by smoothing the volumetric part of the stress tensor (i.e. the pressures). This is easily seen when considering the averaged nodal pressure (ANP) approach used by Hamad [5] to smooth the pressures. In the ANP approach the stress tensor is evaluated as:

$$\boldsymbol{\sigma} = -\bar{p}\mathbf{I} + \boldsymbol{\tau} \tag{2.35}$$

with $\boldsymbol{\tau}$ the deviatoric stress and the $\bar{p}$ the isotropic part (the pressure), computed as [40]:

$$\bar{p} = \frac{1}{n}\sum_{i=1}^{n} p_i \tag{2.36}$$

in which n denotes the number of vertices attached to element e.
Note that such an averaging approach results in a piecewise-constant pressure defined at the element center, thus effectively resulting in a P_1P_0 type element [4]. Hamad [5] maps the cell averaged pressures back to the nodes as:

$$\bar{p}_i = \frac{\sum_{e_i} \bar{p}\Omega_{e_i}}{\sum_{e_i} \Omega_{e_i}} \tag{2.37}$$

in which the summation e_i runs over the elements e attached to node i. The whole procedure can thus be considered to be a smoothing of the noisy pressure field and hence a rather ad-hoc method for improving the pressure fields.
An important decision to be made in MPM is the choice for the basis function $N(\mathbf{x})$. Complicating factor in this is that instead of using the optimal Gauss points (as in regular FEM), the particle positions are used to perform the numerical integration in MPM. This MPM concept is further explained in Section 3.1. Here it is only noted that such an approach is inherently more noisy than optimal quadrature rules as used in FEM. Another immediate result of using particles as integration points is that it becomes practically impossible to use standard higher order elements because the corresponding basis functions (such as the quadratic shape function) may have negative values in their domain of support. When performing the numerical integration at particle positions, this can lead to unphysical negative masses at certain grid points [41]. Amongst others, such as ease of implementation and the partition of unity property, it is for this reason that the linear elements are most often used in hybrid methods, although the application of B-splines is an active field of research ([41], [42]). This type of basis function has the attractive properties of being strictly positive and at least C^1 continuous and satisfying partition of unity. It is therefore possible to obtain a smoother representation of field quantities and their spatial derivatives using these shape functions [41]. Implementation, especially on unstructured meshes, is however less straightforward.

[4] An explanation of this notation is presented in Section 3.2

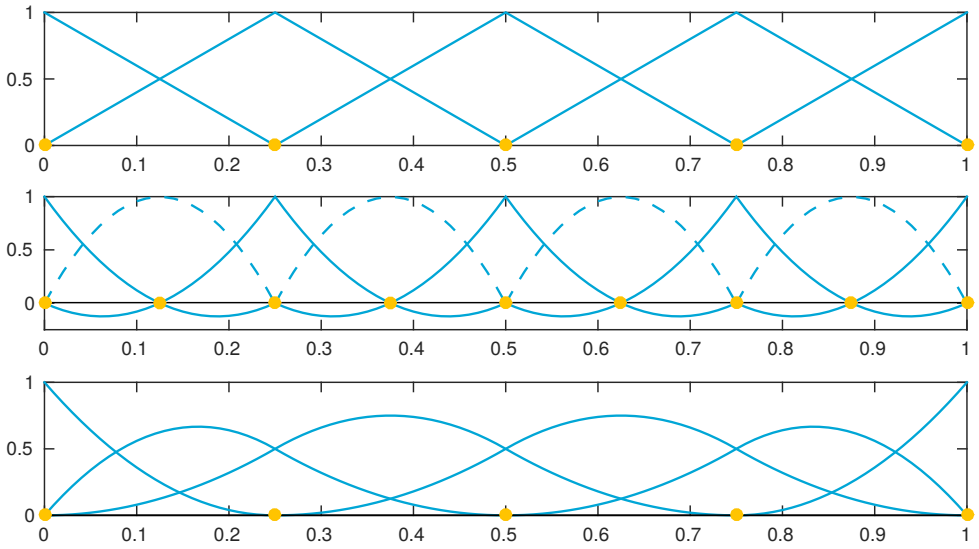


Figure 2.4: Linear (top), quadratic (middle), and quadratic B-spline (bottom) basis functions. FEM nodes are highlighted in blue. Note that the quadratic basis function is negative in part of its support domain.

2.4 Issues in hybrid particle methods

A few issues are encountered in the hybrid particle-mesh methods. Understanding the behavior of particle mesh methods therefore requires to treat the different issues briefly.

A first fundamental issue to be mentioned is the so-called 'ringing instability'. The phenomenon was already recognized by the developers of PIC, although an extensive description and analysis was first given by Brackbill [24]. Occurence of this instability is due to the different amount of degrees of freedom represented by the particles and those represented by the grid. In other words, since the number of particles generally exceeds the number of grid points, the particle resolution is much higher than the grid resolution. This resolution difference may give rise to a phenomenon termed 'aliasing' which means that two particle states can result in the same sampled information at the background grid. A simple illustration of this 'undersampling problem' is given in Figure 2.5.

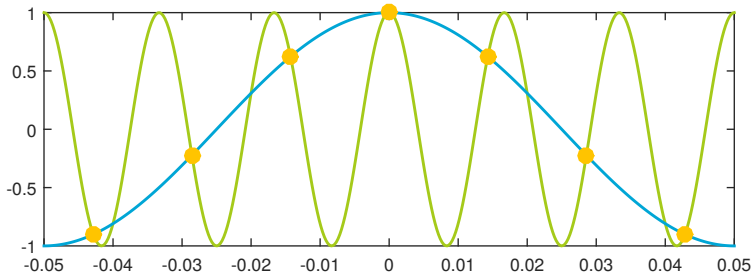


Figure 2.5: Principle of aliasing: the high frequency line (green) is indistinguishable from the low frequencey line (blue) when sampled at the positions of the yellow dots.

The phenomenon is especially pronounced in stagnating flows, leading to particle clumping and disordering [26].

In [24] it was shown that a proper choice for the assignment function (used for the back and forth mapping of particle and grid data) can mitigate the ringing instability. Gritton [27] relates the ringing instability to the non-trivial null space of the particle-to-grid mapping matrix and developed a filtering procedure for simple problems. Although the ringing instability is observed in all particle methods, it can be anticipated that it is especially prounounced in the full particle methods (FLIP and MPM) [16]. Reason for this is

that PIC forces all information through the grid, thus acting as a filter to the particle data. On the other hand, in FLIP and MPM, particle data is preserved between consecutive time steps, and thus instabilities may persist and grow unboundedly over multiple time steps [16]. To oppress the particle noise, a blend between a PIC and FLIP update is often used for updating the particle velocities, i.e. the particle velocity is updated as a combination of Eq. (2.29) and Eq. (2.31) (see e.g. [6], [18], [32]).
A second issue, which got especially much attention in the MPM community, is the grid-crossing error. This error occurs when particles cross an element edge, suddenly leading to a different distribution of nodal masses [10]. However, the lack of C^1 continuity of the shape functions is mentioned to be the most important source for the grid-crossing error ([35], [42], [43]). When using C^0 continuous shape functions, the gradient of the shape function is discontinuous between elements. This discontinuity leads to a sudden change in the particle contribution to the internal force (computed with Eq. (2.32)) when a particle crosses the boundary. This is visualized in Figure 2.6.

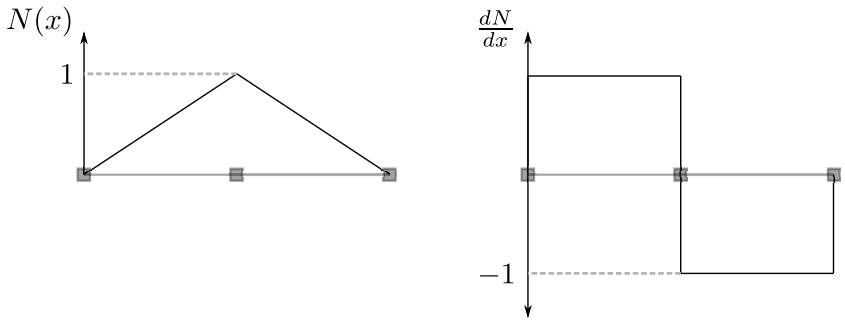


Figure 2.6: The jump in derivative of C^0 continous functions.

Increasing the number of particles per element will mitigate the problem, however, the only way to circumvent this problem is to use shape functions being at least C^1 continuous, which implies the shape function gradient to be continuous between neighboring elements. It is for this reason that the Generalized Interpolation Material Point (GIMP) method was introduced [43]. The key in understanding this method is that particles are not just points as in regular MPM but physical volumes having a weight function. This results in a smeared contribution of the particles to field variables. Another research direction is using B-splines as basis functions ([42], [44]). As can be seen in Figure 2.4, these shape functions are continuous between elements. However, it can be questioned how useful such shape functions are for multiphase problems, since the smooth character of the shape functions are expected to diffuse material interfaces.
A third issue, which is common to both hybrid particle-mesh methods and fully Lagrangian methods is the particle distribution disorder. Over time the particles slowly drift away from an ideal, regular distribution and tend to clump or separate (see e.g. [14]). As an ultimate result, unphysical holes can occur in the continuum. Different solutions have been proposed in literature to obtain a regular distribution of particles. One of the methods (proposed in [45] and used in e.g. [15] and [18]) is to apply a 'spring force' between the particles. In this approach, particle p is shifted according to:

$$\Delta \mathbf{x}_p = -\Delta t \gamma_s d \sum_{\tilde{p} \in P_w} \frac{\mathbf{x}_{\tilde{p}} - \mathbf{x}_p}{\|\mathbf{x}_{\tilde{p}} - \mathbf{x}_p\|} w(\mathbf{x}_{\tilde{p}} - \mathbf{x}_p, d) \tag{2.38}$$

where γ_s represents a stiffness parameter and w is a weight function with bounded (radial) support d, so in general w is different from the interpolation functions. Moreover, the summation runs over the neighboring particles defined by the set P_w:

$$P_w = \{\tilde{p} \,:\, w(\mathbf{x}_{\tilde{p}} - \mathbf{x}_p) \neq 0\} \tag{2.39}$$

An immediate drawback of this approach is that a particle neighbor list has to be constructed, which can result in high CPU-times [15]. Furthermore, the stiffness parameter γ_s is determined in an ad hoc manner and is problem-specific. In addition, after shifting the particles, the particle properties needs to be updated. Common approach is to interpolate the grid properties to the updated particle positions effectively adding some additional numerical diffusion ([15], [45]).

Finally, it should be mentioned that hybrid particle-mesh methods are still in their infancy. Numerical behavior of the methods is therefore far from being understood, many of the approaches seen so far contain therefore some ad hoc steps in order to obtain 'realistic' results.

2.5 Summary and conclusions

In short it can be said that hybrid particle-mesh methods attempt to combine the advantages of Eulerian and Lagrangian methods while avoiding the disadvantages of both approaches. On the one hand, hybrid methods are capable of tracking material interfaces and the non-linear advection term, present in fully Eulerian methods, is conveniently avoided. Nevertheless, hybrid methods can still benefit from typical grid-based approaches such as using fast solvers and enforcing the incompressibility constraint and boundary conditions in a convenient manner. Moreover, since a well-defined background grid is used to take the particle interactions into account, expensive particle operations (such as neighbor searching) are avoided. Hybrid methods thus attempt to formulate each step in the most convenient representation. As a consequence, information has to be transferred between different representations. This information transfer is far from being trivial and in fact the key difference between some known hybrid methods. In the original PIC the only property stored at particle level was the mass, thus PIC could be more or less regarded as a grid method augmented with particles for the advection of field variables. In FLIP (or full-particle PIC) some responsibilities were shifted to the particles, so that all information is stored at particle level, resulting in particles to be the representation of the continuum. MPM takes this even further by evaluating numerical integrals at particle positions and using particle related weights.
As explained in this chapter, the original PIC is known to be very diffusive. Therefore, only FLIP or MPM type methods will be considered in this thesis. As main drawbacks of these methods the aliasing or undersampling problem was mentioned, as well as the grid-crossing error and the particle disorder. And so, although hybrid methods appear to be feasible and efficient methods for simulating free surface flows, some fundamental issues are encountered in this class of methods.

3

Formulating a hybrid particle-mesh method for fluid flows

Based on the overview of the different hybrid particle-mesh methods from the previous chapter, a generic outline for a hybrid particle-mesh method will be presented in this chapter. Abstractly, it can be said that all the kinematic steps are conducted at particle level, whereas all the dynamic steps are performed at the background grid. To accomplish this, all hybrid methods follow the four basic steps distinguished in the previous chapter:

1. Mapping the state variables to the background grid. So this step can be coined as the 'particle-to-grid mapping'.
2. Solving the governing fluid equations at the background grid.
3. Updating the particle properties given the solution at the background grid, i.e. the 'grid-to-particle mapping'.
4. The particle advection.

These steps are subsequently discussed in the following sections, where special attention is paid to the different model approaches available. As such, primary objective of this chapter is to get an understanding of the relationship between grid and particles and how different model implementations will influence this relationship. In other words, this chapter attempts to answer the question: what does it mean to be hybrid?

3.1 Particle-to-grid mapping

First step in the hybrid particle-mesh methods is the mapping of state variables to the background grid. As discussed in the previous chapter, the only state variable to be mapped in the original PIC is the mass, whereas in both FLIP and MPM the mapping includes both a mass mapping and a momentum mapping (and, when required, also an energy mapping). The main questions related to the mapping of the state variables is: 'when' to map and 'how' to map these variables.
Concerning the question how to map, the most common answer is: use a mapping such that the grid representation of a mapped quantity equals the particle representation of the physical quantities. In this sense one can speak of a 'conserving mapping scheme'. And so, restricting the discussion to the conservation of mass and linear momentum, the particle-to-grid mapping must ensure:

$$\int_{\Omega} \rho \mathrm{d}\Omega = \sum_{p \in P_{\Omega}} m_p \tag{3.1}$$

$$\int_{\Omega} \rho \mathbf{u} \mathrm{d}\Omega = \sum_{p \in P_{\Omega}} m_p \mathbf{u}_p \tag{3.2}$$

with m_p the particle mass and $\mathbf{u}_p$ the particle velocity. Furthermore, the left hand side represents the grid representation of the state variables and the summation at the right hand side runs over the set of particles in the domain Ω, thus representing the particle representation of the state variables, i.e.:

$$P_\Omega = \{p \,:\, \mathbf{x}_p \in \Omega\} \tag{3.3}$$

Such a mapping is easily achieved by replacing the integrals with a summation over the particles. Those integrals appear for example in the Galerkin weak formulation of the equations. Derivation of these equations will be discussed in the next section, here it is only noted that mass related integrals will take the form [4]:

$$\int_\Omega (\bullet)\rho \mathrm{d}\Omega \cong \sum_p (\bullet)_p m_p \tag{3.4}$$

As an example, consider the mass matrix which appears when discretizing the dynamic equations. In the FEM formulation, this matrix will take the form (at element level):

$$(M_e)_{ij} = \int_{\Omega_e} \rho N_i N_j \mathrm{d}\Omega \tag{3.5}$$

After substituting in Eq. (3.4), this becomes:

$$(M_e)_{ij} = \sum_{p \in P_e} N_i(\mathbf{x}_p) N_j(\mathbf{x}_p) m_p \tag{3.6}$$

where $N_i(\mathbf{x}_p)$ are the finite element basis functions associated with node i and evaluated at position $\mathbf{x}_p$. Furthermore, the summation runs over the set of particles in element e:

$$P_e = \{p \,:\, \mathbf{x}_p \in \Omega_e\} \tag{3.7}$$

Indeed, this procedure can be interpreted as using the discrete particles as (moving) integration points. Therefore, the particle distribution plays an important role and particles are a true representation of the continuum. The pros and cons of using such a particle based integration scheme are:

- The advantage of using the particles as integration points is that it is relatively straightforward to construct a mapping which is both mass and momentum conserving, in the sense that the grid mass and momentum are equal to the particle mass and momentum. In fact it was shown by Burgess *et al.* [46] that angular momentum and kinetic energy are also conserved in the mapping between particles and grid, using the approach outlined above.
- Another advantage is that the particle distribution is accounted for in the computation as the particle concentration does immediately influence the mass distribution at the background grid. It can be anticipated that such a mechanism does effectively control the particle distribution.
- A clear drawback in such an approach is however that it will introduce noise. This can be attributed to the non-optimal point-wise quadrature: a slightly different particle distribution will lead to a different distribution of nodal masses and nodal forces and will thus give rise to (spurious?) particle motions. This problem is identified in MPM literature and sometimes mitigated by using an optimal quadrature rule for some terms (e.g. the internal body force term) in fully filled elements [35].

The latter issue can also be mitigated by using basis functions with an extended domain of support such as B-splines as suggested by, e.g. Andersen and Andersen [41] and Steffen *et al.* [42]. It is nevertheless noted that irrespective of the basis functions, a non-optimal quadrature is obtained when using particle positions as the quadrature weights.
An alternative approach for the latter mentioned drawback, which appears to be used in the graphics community (see e.g. [6], [18]) as well as geophysical problems [47], is to use a weighted least squares approximation to *sample* the data at the background grid from the particle data. Theory behind weighted least square approximations can be found in [48]. The key idea behind the mapping is to minimize the

weighted difference between the local approximation of the state variable (given by the scattered points) and the function, that is, minimize the functional:

$$J = \sum_{p \in P_w} w(\mathbf{x} - \mathbf{x}_p) \left(f(\mathbf{x}) - f(\mathbf{x}_p) \right)^2 \tag{3.8}$$

where the summation runs over the particles in the domain of support of the weight function w, that is:

$$P_w = \{ p \ : \ w(\mathbf{x} - \mathbf{x}_p) \neq 0 \} \tag{3.9}$$

Typical weight function shapes are to be discussed next. Furthermore, the function $f(\mathbf{x})$ in the functional is approximated as:

$$f(\mathbf{x}) \approx f_h(\mathbf{x}) = \sum_{i=1}^{m} q_i(\mathbf{x}) a_i(\mathbf{x}) \equiv \mathbf{q}^T(\mathbf{x}) \mathbf{a}(\mathbf{x}) \tag{3.10}$$

with m the order of the monomial $q_i(\mathbf{x})$ with coefficients $a_i(\mathbf{x})$. In 2D, the linear basis $\mathbf{q}(\mathbf{x})$ is for example given by:

$$\mathbf{q}^T(\mathbf{x}) = [1, x, y] \qquad\qquad m = 3 \tag{3.11}$$

Upon inserting the approximation for $f(\mathbf{x})$, Eq. (3.8) can be rewritten as:

$$J = (\mathbf{Q}\mathbf{a} - \mathbf{f})^T \mathbf{W}(\mathbf{x}) (\mathbf{Q}\mathbf{a} - \mathbf{f}) \tag{3.12}$$

with:

$$\mathbf{f}^T = (f_1, f_2, ..., f_k) \tag{3.13}$$

$$\mathbf{Q} = \begin{bmatrix} q_1(\mathbf{x}_1) & q_2(\mathbf{x}_1) & \cdots & q_m(\mathbf{x}_1) \\ q_2(\mathbf{x}_2) & q_2(\mathbf{x}_2) & \cdots & q_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ q_1(\mathbf{x}_k) & q_2(\mathbf{x}_k) & \cdots & q_m(\mathbf{x}_k) \end{bmatrix} \tag{3.14}$$

$$\mathbf{W}(\mathbf{x}) = \begin{bmatrix} w(\mathbf{x} - \mathbf{x}_1) & 0 & \cdots & 0 \\ 0 & w(\mathbf{x} - \mathbf{x}_1) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w(\mathbf{x} - \mathbf{x}_k) \end{bmatrix} \tag{3.15}$$

$$k = |P_w| \tag{3.16}$$

In these matrices, k is the number of data points (read: particles) for which the weight function is not equal to zero, m is the number of points where an interpolated value is requested (read: background grid nodes). Furthermore, the functions $w(\mathbf{x})$ are weight functions. These functions typically have the following characteristics: 1) they are strictly positive and 2) their value should tend to zero when $\| \mathbf{x} - \mathbf{x}_p \|$ increases. In practice, the latter condition ofen lead to weight functions with local support, that is, if $\| \mathbf{x} - \mathbf{x}_p \|$ exceeds a certain threshold, the value of the weight function is set to 0.
The least square fit is found by the extremum of J with respect to $\mathbf{a}(\mathbf{x})$, resulting in [48]:

$$\mathbf{A}(\mathbf{x})\mathbf{a}(\mathbf{x}) - \mathbf{B}(\mathbf{x})\mathbf{f} = 0 \tag{3.17}$$

with matrices $\mathbf{A}(\mathbf{x})$ and $\mathbf{B}(\mathbf{x})$ are respectively given by:

$$\mathbf{A}(\mathbf{x}) = \mathbf{Q}^T \mathbf{W}(\mathbf{x}) \mathbf{Q} \tag{3.18}$$

$$\mathbf{B}(\mathbf{x}) = \mathbf{Q}^T \mathbf{W}(\mathbf{x}) \tag{3.19}$$

Hence, the monomial coefficients can be expressed as:

$$\mathbf{a}(\mathbf{x}) = \mathbf{A}^{-1}(\mathbf{x}) \mathbf{B}(\mathbf{x}) \mathbf{f} \tag{3.20}$$

Upon substituting this in Eq. (3.10), yields:

$$f_h(\mathbf{x}) = \sum_{p \in P_w} \phi_p(\mathbf{x}) f_p \tag{3.21}$$

with the moving weighted least square shape functions given by:

$$\boldsymbol{\phi}(\mathbf{x}) = \mathbf{q}^T(\mathbf{x}) \mathbf{A}^{-1}(\mathbf{x}) \mathbf{B}(\mathbf{x}) \tag{3.22}$$

The least square mapping presented above can be regarded as a 'best-fit' of the state variable defined at particle level onto a state variable field defined at the background grid. Using this approach, the background grid can still be used for the optimal numerical integration. Some other important properties to keep in mind when using a weighted least square approach are:

- The shape of the moving least square basis functions depends on the choice for the weight function $w(\mathbf{x})$. The standard polynomial finite element basis functions $N_i(\mathbf{x})$ are for example obtained when setting the weight functions $w(\mathbf{x})$ to piecewise constants over each subdomain or element [49].
- The moving least squares technique can be regarded as a generalization of the MPM approach. As such, it offers more flexibility compared to MPM (e.g. by choosing the weight functions). As will be shown in the sequel, the flexibility of the weighted least square mapping allows for an implementation employing optimal quadrature schemes at the background grid.
- This feature can come however at the expense of:
  - The risk of losing particle information, such as information about the particle concentration (although it should be noted that this can be controlled with the weighting function).
  - In its generic form, the least square approach does not automatically lead to a formulation in which the mapping ensures the particle and the grid representation of physical quantities to be the same. Big question of course is whether this is will be an issue.
- Indeed, depending on the specific least square implementation for the particle-to-grid mapping, responsibilities can be shifted from background grid to the particles and vice versa. In the least square approach leading to MPM, the particles are for example used as moving integration points, whereas another least square approach (wich will be used in this thesis) consider the grid points to be sample points of the continuum which is represented by a set of particles [50]. In the latter approach, more emphasis is put on the background grid, as it proceeds by approximating the integrals at the background grid, using optimal quadrature rules.

It is to be mentioned that the least square concept is extensively used in some meshless methods, such as the Element Free Galerkin Method [49] as well as particle-in-cell methods (see e.g. [50], [51]). A particularly simple implementation of a least square approximation which still allows for optimal background grid quadrature is given in the next chapter.
A first important lesson following from the particle-to-grid mapping procedure is that a mass and momentum conserving mapping as used in MPM can easily lead to a noisy behavior. Main reason for this is the non-optimal quadrature by using the particles as integration points. This can be mitigated by using e.g. B-splines. An alternative and more flexible approach can be found in using least square approximations in which a background grid field is constructed from particle data. Flexibility of this method is found in the choice for the weight function, furthermore, an optimal quadrature scheme can still be used at the background grid. Nevertheless, it can be questioned whether:

- the grid representation of physical quantities is equal to the particle representation (e.g. whether the grid mass equals the particle masses (Eq. (3.1))).
- the least square approach fully exploits the particle information (e.g. particle distribution).

The second question to be answered was: when to map the data to the background grid. Again, different options are available:

- Map the state variables (mass and velocity/momentum) at the beginning of each time step. Evaluate the equations at the background grid.

- Perform some of the computational steps at the particle level and map the updated particle properties to the grid. Evaluate the equations at the background grid given the updated particle properties.

These options are for example relevant when mapping the particle velocity. Either the true particle velocity $\mathbf{u}_p$ is mapped, body forces and viscous forces are applied at the background grid. Another option is to map an updated velocity from the particles to the background grid, in which the body forces and viscous forces are already incorporated. The latter option is however not further investigated in this thesis.
It is clear that there are many different options in the 'particle-to-grid' mapping procedure, too many to deal with in a single thesis. Therefore the following choices are made:

- The state variables are approximated using a least square mapping given by a specific implementation of Eq. (3.21) such that an optimal grid-quadrature can be used. Furthermore, gradients are evaluated at the background grid.
- Relevant quantities (velocity and density) are mapped at the beginning of each time step. This means that the configuration of the grid is only important within a single timestep, an updated or completely new grid can be used for the next timestep although this requires some additional particle administration. This feature of hybrid methods can be mentioned to be one of the advantages of hybrid particle methods over standard Eulerian methods ([33]; [35]).

Ramifications of these choices should be kept in mind. Probably the most important consequence is that the mapping is not automatically mass and momentum conserving. Realizing that the way of transferring particle data to the grid highly determines the importance which is assigned to grid and particles, it is noted that the chosen approach assigns relatively much importance to the grid.

3.2 Solving the grid equations

After mapping the state variables, the equations at the background grid are solved. One of the advantages of hybrid methods over fully Lagrangian methods is that dedicated grid solvers can be used for this step. In PIC and FLIP, the governing equations are usually discretized using a Finite Difference or Finite Volume approach, whereas in MPM the equations are discretized in a finite element (FEM) form. In this thesis, a FEM approach will be exploited as it is anticipated that this method offers more flexibility (for example in element choice) compared to the other grid-based discretization techniques. This choice does however certainly *not* mean that the chosen approach is an MPM approach, instead, in the previous section it was motivated to evaluate gradients and perform the numerical quadrature at the background grid. Those steps are instead typically performed at particle level in the MPM approach.

3.2.1. The weak form and the Galerkin method

Starting point for the discussion are the incompressible Navier-Stokes equations, given by Eq. (2.13) and Eq. (2.6):

$$\rho \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} = \mathbf{f} - \nabla p + \nabla \cdot \boldsymbol{\sigma} \qquad \text{in} \quad \Omega \times I \tag{3.23}$$

$$\nabla \cdot \mathbf{u} = 0 \qquad \text{in} \quad \Omega \times I \tag{3.24}$$

in which the viscous stress tensor $\boldsymbol{\sigma} = 2\mu\nabla^s\mathbf{u}$ and $\mathbf{f}$ the body force. Furthermore, the fluid domain is denoted by $\Omega \subset \mathbb{R}^d$ with d the spatial dimension and the time interval denoted as $I = (t^0, t^N)$.
In this coupled set of equations, the momentum balance is represented by Eq. (3.23) while the mass balance reduces for incompressible fluids to Eq. (3.24), stating that the divergence of the velocity field $\mathbf{u}$ equals 0 for an incompressible fluid flow. Furthermore, it is noted that the problem needs to be completed by specifying proper boundary and initial conditions (see Section 2.1.2).
First step in the finite element approach is to derive a weak form. This is achieved by multiplying Eq. (3.23) and Eq. (3.24) with the weight functions $\mathbf{v} \in \mathcal{V}$ and $q \in \mathcal{Q}$ respectively and integrating over the domain Ω. Furthermore, it is supposed that $\mathcal{V} \subset \mathbf{L}^2(\Omega)$ and $\mathcal{Q} \subset H^1(\Omega)$, in which the $L^2(\Omega)$ space consists of the set of square integrable functions, and the $H^1(\Omega)$ space consists of the set of functions having also square-integrable derivatives. Clearly, it holds that $H^1 \subset L^2$.
Upon performing the above described steps, the resulting equations become:

$$\int_\Omega \rho \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} \cdot \mathbf{v} \mathrm{d}\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} \mathrm{d}\Omega - \int_\Omega \nabla p \cdot \mathbf{v} \mathrm{d}\Omega + \int_\Omega (\nabla \cdot \boldsymbol{\sigma}) \cdot \mathbf{v} \mathrm{d}\Omega \qquad \forall \mathbf{v} \in \mathcal{V} \tag{3.25}$$

$$\int_\Omega (\nabla \cdot \mathbf{u})\, q \mathrm{d}\Omega = 0 \qquad \forall q \in \mathcal{Q} \tag{3.26}$$

For the time being, the term involving the viscous shear stresses will be dropped. In Section 4.3 an implementation dependent formulation of this term will be presented. By leaving out the viscous term, the resulting set of equations reduce to the incompressible Euler equations.
Integration by parts is applied on the term involving the divergence of the velocity. Thus, the weak form becomes: find $\mathbf{u} \in \mathbf{L}^2(\Omega)$ and $p \in H^1(\Omega)$ such that:

$$\int_\Omega \rho \frac{\mathrm{D}\mathbf{u}}{\mathrm{D}t} \cdot \mathbf{v} \mathrm{d}\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v} \mathrm{d}\Omega - \int_\Omega \nabla p \cdot \mathbf{v} \mathrm{d}\Omega \qquad \forall \mathbf{v} \in \mathcal{V} \tag{3.27}$$

$$\int_\Omega \mathbf{u} \cdot \nabla q \mathrm{d}\Omega = \oint_{\partial\Omega} \mathbf{u} q \cdot \mathbf{n} \mathrm{d}\Gamma \qquad \forall q \in \mathcal{Q} \tag{3.28}$$

Where it is anticipated that the boundary integral term in Eq. (3.28) will follow from the prescribed boundary velocity.
These equations have to be solved for the primitive variables $\mathbf{u}$ and p. The essence in the finite element method is to seek approximate solutions for these variables. In order to achieve this, finite dimensional function spaces $\mathcal{V}_h \subset \mathcal{V}$ and $\mathcal{Q}_h \subset \mathcal{Q}$ are defined:

$$\mathcal{V}_h = \{\mathbf{v}_h \in \mathbf{L}^2 : \mathbf{v}_h \in \mathbf{P}^k(\Omega_e) \ \ \forall e\} \tag{3.29}$$

$$\mathcal{Q}_h = \{q_h \in H^1 : q_h \in P^l(\Omega_e) \ \ \forall e\} \tag{3.30}$$

in which Ω_e is the collection of finite elements in the domain Ω, i.e.:

$$\Omega = \cup_e \Omega_e \tag{3.31}$$

and $\mathbf{P}^k$ and P^l are polynomial basis functions, with $k \geq 0$ and $l \geq 1$.
In the Galerkin procedure, the approximate solutions or trial functions $\mathbf{u}_h$ and p_h come from the same space as the discrete weight functions $\mathbf{v}_h \in \mathcal{V}_h$ and $q_h \in \mathcal{Q}_h$. The only difference between the trial and the weight functions is that trial functions have to satisfy the *inhomogeneous* Dirichlet boundary conditions, whereas the weight functions satisfy the *homogeneous* Dirichlet boundary conditions at these locations.
Upon substituting the discrete trial and test functions in the weak form Eqs. (3.27)-(3.28), the semi-discrete problem is given by: find $\mathbf{u}_h \in \mathcal{V}_h$ and $p_h \in \mathcal{Q}_h$ such that:

$$\int_\Omega \rho \frac{\mathrm{D}\mathbf{u}_h}{\mathrm{D}t} \cdot \mathbf{v}_h \mathrm{d}\Omega = \int_\Omega \mathbf{f} \cdot \mathbf{v}_h \mathrm{d}\Omega - \int_\Omega \nabla p \cdot \mathbf{v}_h \mathrm{d}\Omega \qquad \forall \mathbf{v}_h \in \mathcal{V}_h \tag{3.32}$$

$$\int_\Omega \mathbf{u}_h \cdot \nabla q_h \mathrm{d}\Omega = \oint_{\partial\Omega} q_h \mathbf{u}_D \cdot \mathbf{n} \mathrm{d}\Gamma \qquad \forall q_h \in \mathcal{Q}_h \tag{3.33}$$

in which $\mathbf{u}_D$ follows from the boundary condition defined at $\partial\Omega$.
Furthermore, defining the finite dimensional function spaces by assigning to each element Ω_e a set of $P(\Omega_e)$ elementary basis functions as done in Eqs. (3.29)-(3.30), the domain integrals can be decomposed in a summation of the element domain integrals, i.e.:

$$\int_\Omega (\bullet) \mathrm{d}\Omega = \sum_e \int_{\Omega_e} (\bullet) \mathrm{d}\Omega \tag{3.34}$$

3.2.2. Time integration

In order to obtain a fully discrete set of equations, the time domain I is splitted in a discrete number of time levels $I = (t^0, t^1, ..., t^N)$ with associated sub-intervals $\Delta t = (t^n, t^{n+1})$. The time derivative in Eq. (3.32) can thus be approximated as:

$$\frac{\mathrm{D}\mathbf{u}_h}{\mathrm{D}t} = \frac{\mathbf{u}_h^{n+1} - \mathbf{u}_h^n}{\Delta t} \tag{3.35}$$

Where it is noted that every timestep, a solution has to be found for $\mathbf{u}_h^{n+1}$ and p_h^{n+1} given the solution at time level t^n.

Expressing the primitive variables in terms of the finite element basis functions, an approximation for $\mathbf{u}$ and p at time level t^n becomes:

$$\mathbf{u}_h^n(\mathbf{x}) = \sum_{i=1}^{n_{\text{vn}}} N_i(\mathbf{x})\mathbf{u}_i^n \tag{3.36}$$

$$p_h^n(\mathbf{x}) = \sum_{i=1}^{n_{\text{pn}}} \tilde{N}_i(\mathbf{x})p_i^n \tag{3.37}$$

in which N and $\tilde{N}$ are the basis functions for the velocity and the pressure respectively and $\mathbf{u}_i^n$ and p_i^n nodal degrees of freedom. Furthermore, the summation runs over either the number of velocity nodal unknowns n_{vn} or the number of pressure nodal unknowns n_{pn}. Note that the basis functions for the pressure and the velocity are not necessarily the same, in other words, the pressure and the velocity space are not necessarily similar.

Upon substituting these expressions in the semi-discrete form Eqs. (3.32)-(3.33), the matrix formulation for the fully discrete form of the background grid equations becomes:

$$\frac{1}{\Delta t}\mathbf{M}(\mathbf{U}_k^{n+1} - \mathbf{U}_k^n) + \mathbf{G}_k\mathbf{P}^{n+1} = \mathbf{b}_k^{n+1} \tag{3.38}$$

$$\mathbf{G}_k^T\mathbf{U}_k^{n+1} = \mathbf{h}^{n+1} \tag{3.39}$$

in which the mass matrix $\mathbf{M}$, the discrete gradient operator $\mathbf{G}_k$, the body force vector $\mathbf{b}_k^{n+1}$ and the boundary vector $\mathbf{h}^{n+1}$ are respectively defined as:

$$M_{ij} = \int_\Omega \rho N_i N_j \mathrm{d}\Omega \tag{3.40}$$

$$G_{ij,k} = \int_\Omega N_i(\nabla\tilde{N}_j \cdot \mathbf{e}_k)\mathrm{d}\Omega \tag{3.41}$$

$$b_{i,k} = \int_\Omega \nabla N_i(\mathbf{f} \cdot \mathbf{e}_k)\mathrm{d}\Omega \tag{3.42}$$

$$h_j = \oint_{\partial\Omega} \tilde{N}_j\mathbf{u}_D \cdot \mathbf{n}\mathrm{d}\Gamma \tag{3.43}$$

where the k^{th} component is defined as $k = 1, 2, ..., d$ with d the spatial dimension. The vector $\mathbf{e}_k$ correspondingly denotes the unit vector in Cartesian direction k. Furthermore the vector $\mathbf{U}_k$ denotes the vector of nodal velocity unknowns in direction k and $\mathbf{P}$ the vector of nodal pressures. As a final remark regarding the above presented matrices and vectors it is noted that the volume integral can be replaced by a summation over the element volumes (Eq. (3.34)).

The resulting system can thus be rewritten as:

$$\begin{bmatrix} \mathbf{M}\Delta t^{-1} & \mathbf{G}_k \\ \mathbf{G}_k^T & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{U}_k^{n+1} \\ \mathbf{P}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_k^* \\ \mathbf{h}^{n+1} \end{bmatrix} \tag{3.44}$$

with $\mathbf{b}_k^*$ given by:

$$\mathbf{b}_k^* = \mathbf{b}_k^{n+1} + \mathbf{M}\Delta t^{-1}\mathbf{U}_k^n \tag{3.45}$$

3.2.3. Element considerations

The choice for an element is of crucial importance in the FEM procedure. A finite element mesh typically consists of triangles and quadrilaterals in $\mathbb{R}^2$ and tetrahedrals or hexahedrons in $\mathbb{R}^3$. With the mesh, the domain Ω is partitioned in a set of finite elements:

$$\Omega = \cup_e \Omega_e \tag{3.46}$$

Choosing triangular or tetraheral shaped elements, it is possible to construct a mesh following irregularly shaped domains. Moreover, with these element shapes it is possible to vary the spatial resolution of the mesh, thus allowing for finer meshes in regions where sharp gradients are expected while keeping the mesh coarse in areas of little interest.
After generating the mesh, a choice for the finite element basis functions has to be made. In Eq. (3.29) and Eq. (3.30) it has already been assumed that these polynomial basis functions have local support, i.e. they are defined element wise. The order of the polynomial is determined by k and l. The basis functions $N_i(\mathbf{x})$ have the characteristic feature of being one in the corresponding node i at the background grid and being zero in all other nodes.
For incompressible fluid simulations, the order k for the velocity basis functions $\mathbf{P}^k(\Omega_e)$ and the order l for the pressure basis functions $\mathbf{P}^l(\Omega_e)$ cannot be chosen independently. The reason for this becomes clear when investigating the system of equations to solve, Eq. (3.44). This system has the structure of a saddle-point problem. Difficulty in solving these type of systems is that the number of pressure unknowns assembled in $\mathbf{P}$ may not exceed the number of velocity unknowns, otherwise the resulting system becomes inconsistent or singular. The problem of volumetric locking, which is observed in the literature on MPM for weakly compressible fluid flows (see e.g. [4] and [5]), is due to this problem as the same finite element space is used for both the velocity and the pressure approximation without applying additional stabilization techniques.
The locking problem can be avoided by choosing the proper combination of velocity and pressure function spaces. A sufficient condition for choosing these spaces is the Ladyzhenskaya-Babuska-Brezzi (LBB) condition. Discussion of this condition is out of the scope of the thesis, but the important lesson to keep in mind is that the function spaces for the pressure and the velocity (given by $\mathcal{V}_h$ and $\mathcal{Q}_h$ respectively) are coupled and cannot be chosen independently. In practice, this means that the element basis functions should be chosen with care. Elements satisfying the LBB condition are known as 'admissible elements'.
A good introduction on the choice of the finite element when applied to incompressible fluid flows is given by Gresho and Sani [52]. Only a few important concepts which are required to understand the remainder of this thesis, are summarized here:

- Finite elements can be categorized in different groups. An important distinction is made between the equal-order elements and the mixed-interpolation elements. In the first category, the basis functions for velocity and pressure are the same (i.e. $N = \tilde{N}$). Usually, stabilization techniques are required in order to solve the saddle-point problem Eq. (3.44) using equal order elements.
  When the velocity basis functions are different from the pressure basis functions, the element is labeled as a mixed-interpolation element.
- Since there exists a large variety of finite elements an efficient terminology has been developed to classify the elements. For triangular elements and tetrahedral elements the notation $\mathrm{P}_m\mathrm{P}_n$ is used, for quadrilaterals the notation $\mathrm{Q}_m\mathrm{Q}_n$ is used. In this notation the subscript m means that the velocity is approximated by piecewise, complete polynomials of order m while n denotes the polynomial degree for the pressure basis functions. A special class of elements are the enriched elements, in wich a kind of 'bubble function' is added to the velocity space. This is denoted with P_m^+ or Q_m^+.

Two popular mixed finite elements are the $\mathrm{P}_0\mathrm{P}_1$and the $\mathrm{P}_1^+\mathrm{P}_1$element, Figure 3.1

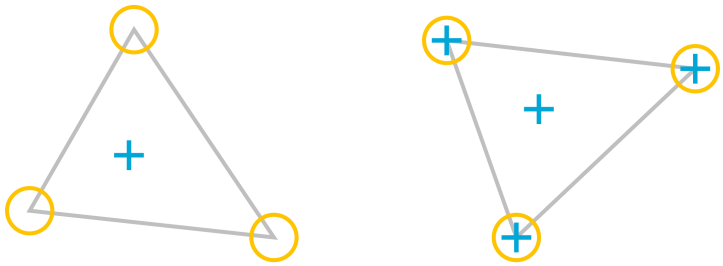


Figure 3.1: The P_0P_1(left) and the $P_1^+P_1$element (right). Velocity DOFs are denoted with a $+$ sign, pressure DOFs are denoted with a $\circ$.

From this figure it follows that the pressure basis functions for the P_0P_1element are piecewise linears, whereas piecewise constant functions are used for the velocity unknowns. Hence, the finite dimensional function spaces are given by:

$$\mathcal{V}_h = \{\mathbf{v} \in \mathbf{L}^2 : \mathbf{v} \in \mathbf{P}^0(\Omega_e) \ \forall e\} \tag{3.47}$$

$$\mathcal{Q}_h = \{q \in H^1 : q \in P^1(\Omega_e) \ \forall e\} \tag{3.48}$$

Another stable, low order element, is the $P_1^+P_1$element (MINI element [53]). In this element, the piecewise linear polynomial basis for the velocity is enriched with a so-called 'bubble function', being either a piecewise linear or a cubic function. This bubble function has the property of being one in the element barycenter and zero on element boundaries. Collecting the bubble functions in the $\mathcal{B}_h$ function space, the enriched, mixed function space is given by $\mathcal{W} = (\mathcal{V}_h \oplus \mathcal{B}_h) \times \mathcal{Q}_h$, in which:

$$\mathcal{V}_h = \{\mathbf{v} \in \mathbf{H}^1 : \mathbf{v} \in \mathbf{P}^1(\Omega_e) \ \forall e\} \tag{3.49}$$

and $\mathcal{Q}_h$ remains unaltered.

3.2.4. Modifications of the weak form for hybrid particle-mesh methods

Having derived the weak form of the governing equations the question remains: how do the particles influence these equations? In fact, the particle influence is incorporated in the equations by means of the chosen particle-to-grid mapping. Since a least square approach was chosen to approximate the state variables at the background grid, the particle information is incorporated by approximating the velocity and the density at the *old* time level n with the least square approach, i.e.:

$$\mathbf{u}_h^n(\mathbf{x}) = \sum_{p \in P_w} \phi_p(\mathbf{x}) \mathbf{u}_p^n \tag{3.50}$$

$$\rho_h^n(\mathbf{x}) = \sum_{p \in P_w} \phi_p(\mathbf{x}) \rho_p^n \tag{3.51}$$

where the weighted least square basis functions $\phi(\mathbf{x})$ are defined by Eq. (3.22). These basis functions are *not necessarily similar* to the finite element polynomials $N_i(\mathbf{x})$, instead, the shape of $\phi(\mathbf{x})$ is determined by the weight function introduced in Section 3.1.

It is emphasized that in this approach the numerical quadrature of the Galerkin weak forms is performed at the background grid. Thus assigning relatively much importance to the background grid compared to the particles. As noted in the previous section, this is a fundamental difference with MPM where the quadrature is performed at the particle locations, using the particle volume or mass as quadrature weight. As an example, the mass matrix and the discrete gradient matrix respectively become in MPM (at element level):

$$(M_e)_{ij} = \sum_{p \in P_e} \rho_p N_i(\mathbf{x}_p) N_j(\mathbf{x}_p) V_p \tag{3.52}$$

$$(G_{e,k})_{ij} = \sum_{p \in P_e} N_i(\mathbf{x}_p) \left[\nabla \tilde{N}_j(\mathbf{x}_p) \cdot \mathbf{e}_k)\right] V_p \tag{3.53}$$

with V_p the particle volume.
This MPM type approach is not used in this thesis, instead, the first presented option of a least square approach combined with background grid quadrature is chosen in this thesis. However, the applicability of the MPM approach for solving incompressible fluid flow equations should be kept in mind, because it can be a useful option in future research.

3.3 Grid-to-particle mapping

Based on the grid solution obtained with a procedure as presented in the previous section, particle properties needs to be updated. For the incompressible Euler equation, the only quantity to be updated at particle level is the velocity. The common approach, both in FLIP and in MPM is to update the particle velocities by mapping the nodal accelerations to the particles, so:

$$\begin{aligned}\mathbf{u}_p^{n+1} &= \mathbf{u}_p^n + \sum_i N_i(\mathbf{x}_p^n)\mathbf{a}_i \Delta t \\ &= \mathbf{u}_p^n + \sum_i N_i(\mathbf{x}_p^n)\left[\mathbf{u}_i^{n+1} - \mathbf{u}_i^n\right]\end{aligned} \tag{3.54}$$

See also Eq. (2.31).
In fact, it is this step which distinguishes FLIP and MPM from PIC. Because in a PIC setting the particle velocity is overwritten every timestep as (recalling Eq. (2.29)):

$$\mathbf{u}_p^{n+1} = \sum_i N_i(\mathbf{x}_p^n)\mathbf{u}_i^{n+1} \tag{3.55}$$

Using the PIC update given by Eq. (3.55) leads to extremely diffusive behavior. This diffusive behavior is avoided when using the FLIP update defined by Eq. (3.54). However, a full FLIP update can lead to noise at the particle level (as a result of the ringing instability, see Section 2.4).
In order to combine the advantages of the PIC and the FLIP update, the two approaches are often blended (see e.g. [6], [17], [18]). In this approach the particle velocity is updated as:

$$\mathbf{u}_p^{n+1} = \alpha \sum_i N_i(\mathbf{x}_p^n)\mathbf{u}_i^{n+1} + (1-\alpha)\left\{\mathbf{u}_p^n + \sum_i N_i(\mathbf{x}_p^n)\left[\mathbf{u}_i^{n+1} - \mathbf{u}_i^n\right]\right\} \tag{3.56}$$

Clearly the balance between a FLIP and a PIC update is controlled by the α parameter (hereafter called the PIC fraction), where $\alpha = 0$ results in a pure FLIP update and $\alpha = 1$ results in the basic PIC update. Typical values for the α parameter found in literature are in the range of 0 to 0.05 [6]. The magnitude of this parameter is determined somewhat ad hoc, where it is noted that the parameter should be kept as small as possible in order to avoid excessive numerical dissipation.

3.4 Particle advection

Final step in hybrid particle-mesh methods is the advection of the particles. This is done by solving an ordinary differential equation (ODE) of the form:

$$\left.\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t}\right|_t = \tilde{\mathbf{u}}_p(\mathbf{x}_p, t) \tag{3.57}$$

Where two approaches are distinguished in order to compute the particle advection velocity $\tilde{\mathbf{u}}_p(\mathbf{x}, t)$:

- Advect the particles with their own velocity:

$$\tilde{\mathbf{u}}_p(\mathbf{x}, t) = \mathbf{u}_p(\mathbf{x}, t) \tag{3.58}$$

  where $\mathbf{u}_p$ follows from Eq. (3.56).
- Advect the particle in the grid velocity field, evaluated at the particle position.

$$\tilde{\mathbf{u}}_p(\mathbf{x}, t) = \sum_i N_i(\mathbf{x})\mathbf{u}_i(t) \tag{3.59}$$

Similar to all model choices seen so far, the choice for the particle advection again determines how the 'resposibilities' are divided between particles and grid. Whereas the first option results in a more 'particle oriented' method, the second option results in a more 'grid based method'. Although examples of the first approach can be found in literature [54], it is especially the second approach which has become popular in hybrid methods. Reason for this is two-fold:

- Using the particle velocity for the advection can result in multi-streaming which means that particles at the same point in space have a different velocity [24]. To reduce this artifact, a drag term is introduced to account for the difference in particle and fluid velocity. This drag term has a similar function as the artificial viscosity term in SPH, which is also introduced to avoid multistreaming of the particles [26].
- The no-slip condition is automatically satisfied when using a continuous velocity field ($\mathbf{u}_h \in \mathbf{H}^1$). A feature which is beneficial in multi-material or granular media modeling [33].

In this thesis, the grid velocity field will be used for the particle advection, thus again assigning relatively much importance to the background grid compared to the particles.

3.5 Summary and conclusions

In this section a generic outline for a hybrid particle-mesh method is presented. It is clear that the main difficulty in all hybrid particle-mesh methods is how to judge and implement the interaction between the particles and the grid. In fact, it can be argued that the specific implementation of a hybrid particle-mesh approach does result in a method which is entirely grid-oriented (so that the particles are just passive tracers as for example in MAC) or instead entirely particle-oriented (i.e. SPH type methods). It would be too much to cover all different options in one thesis. Based on the discussion in this chapter, the algorithm outline as presented in Algorithm 1 is proposed to use in this thesis. In the next chapter, the implementation of these steps will be discussed.

Algorithm 1 General outline of a hybrid particle mesh method.

1: Initialize grid.
2: Initialize particles on grid.
3: **while** $t < t_{\text{stop}}$ **do**
4: Use a least square mapping, Eqs. (3.50)-(3.51), to project particle properties on the background grid in order to construct:
   - Velocity field
   - Density field
5: Solve Eqs. (3.38)-(3.39) at the background grid for $\mathbf{P}^{n+1}$ and $\mathbf{U}_k^{n+1}$, $k = 1, 2, ..., d$.
6: Update particle properties by means of a grid-to-particle map using Eq. (3.56).
7: Advect particles in the grid velocity field by solving Eq. (3.57).
8: **od**

Four important notes have to be made regarding the model formulation as presented in this chapter:

- By approximating the velocity and the density field according to Eqs. (3.50)-(3.51) and using an optimal quadrature rule at the background grid, the chosen implementation can be regarded to be rather a FLIP implementation than a MPM implementation.
- A FEM approach is used for the discretization of the background grid equations. Given the saddle-point nature of the problem, the pressure and velocity space have to be admissible. This will probably avoid the locking problem often encountered in (weakly compressible) MPM.
- Nodal increments are used for *updating* the particle velocity. This can however lead to noise in the particle velocities, rendering computations inaccurate or unstable. As an 'engineering solution', the FLIP update is blended with a PIC update, thus effectively adding some numerical diffusion. Choosing the blending parameter is a somewhat ad hoc process, and should be topic of future research.
- When using the background grid velocity field for the particle advection as proposed in Section 3.4, the velocity field must be continuous in order to avoid multistreaming of the particles.

4

Model implementation

After formulating a generic outline for a hybrid particle-mesh method for incompressible fluid flows in Chapter 3, the implementation of such a method is discussed in this chapter. This is done by giving a detailed description of the different steps as presented in Algorithm 1. End-product of this chapter is therefore a schematic overview of the implementation of a hybrid particle-mesh method as well as an overview of implementation choices.

4.1 Initializing mesh and particles

4.1.1. Element choice

As discussed in Chapter 3, choosing an element is far from trivial in FEM when solving the incompressible Euler equations or Navier-Stokes equations and according to Chapter 2 it will be even less so for hybrid methods. The main reason for this is that the representation of the particle properties at the background grid have to enter the game somewhere.

To limit the discussion, the focus will be on low order, triangular elements in this thesis. Two examples of such simple, admissible elements were presented in Section 3.2.3. In this thesis, only the P_0P_1element will be considered. Main reasons for choosing this element are the simplicity of the element, while still being admissible. Another distinct advantage of the element is for example the simplicity of the mass matrix, which is even locally invertible due to the discontinuous nature of the velocity basis functions. Finally, since the P_0P_1element is defined on triangular elements in $\mathbb{R}^2$, this element allows to mesh domains of complicated shape and conveniently vary the mesh resolution when desired.

At first sight, the P_0P_1element does not satisfy the LBB condition as the number of pressure nodes at element level exceeds the number of velocity nodes. However, it can be argued that the number of elements quickly exceeds the number of vertices for a growing collection of connected elements and so the number of degrees of freedom for representing the velocity exceeds in general the number of pressure degrees of freedom. Although this remark is far from being a formal proof of the LBB-stability, the element has been successfully applied by e.g. Ando *et al.* [18] in which locking was effectively prevented by using this element. The element is well-suited for the spatial discretization of the Euler equations, although it can be anticipated that discretization of the viscous term in the full Navier-Stokes equations is less straightforward. Because this term is of second-order in the velocity, a regular FEM discretization will require the velocity basis functions to have square-integrable derivatives (i.e. $\mathbf{u}_h \in H^1$). A different approach for the discretization of the viscous term on the P_0P_1element will be discussed in Section 4.3.2.

Another disadvantage of the low-order P_0P_1element is that it will solve for a piecewise constant velocity field. Such a discontinuous velocity field is not suitable for doing the particle advection as it will immediately lead to particle separation when doing the particle advection (Line 7 of Algorithm 1). In order to alleviate this problem, a projection step was proposed by Ando *et al.* [18] in which the centered velocities are mapped to the vertices while keeping the divergence free cell center velocities. When advecting the particles, the elements are temporarily divided in three sub-elements with the cell center position as a shared vertex. This way, a continuous, piecewise linear velocity field is constructed on each sub-element through which

the particles can be advected. The process is schematized in Figure 4.1. Drawback of this approach is that the resulting grid velocity field is not strictly divergence free anymore.

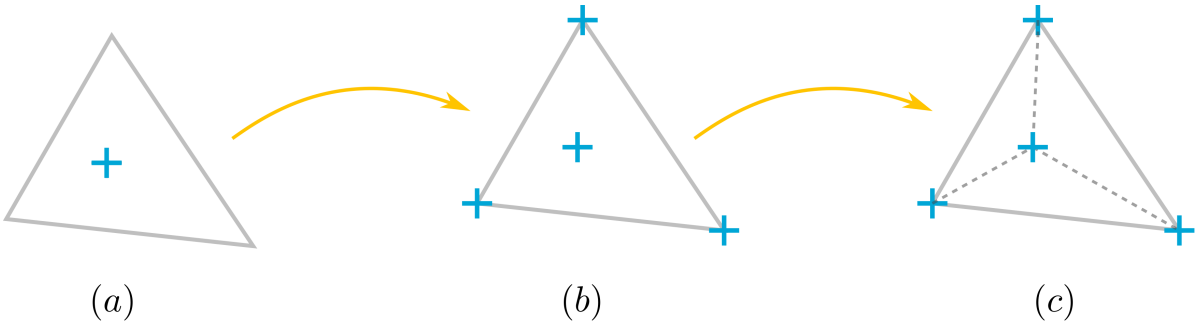


Figure 4.1: Mapping procedure as proposed by Ando *et al.* [18]: (*a*) cell centered velocity field; (*b*) velocity mapped to vertices; (*c*) division into sub-elements.

4.1.2. Mesh generation and particle placement

The mesh is generated using the FEniCS built-in grid generator. A brief description of the FEniCS package can be found in Appendix A. The built-in mesh generator allows for the construction of for example rectangles by means of regular triangular grids of the diagonal and the crossed type, see Figure 4.2. Irregular triangular meshes, as presented in the right panel of Figure 4.2, are generated using GMSH [55]. In the remainder of the report, the different mesh types will be denoted with: 'diagonal', 'crossed' or 'irregular'.

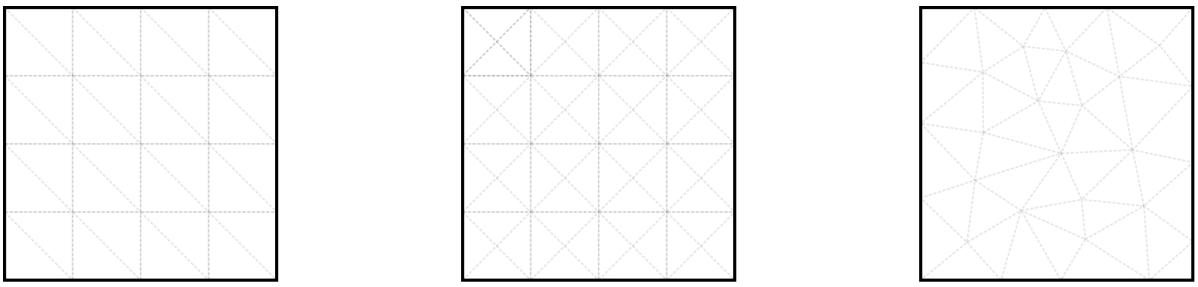


Figure 4.2: Regular triangular mesh of the diagonal-type (left), the crossed-type (middle) and the irregular triangular mesh (right).

Different strategies can be employed for the particle placement, see Figure 4.3. A first approach would be to place the particles at the integration points of the elements. This approach is especially popular in the MPM community, since it reduces the first MPM step to regular FEM provided that the particle volumes correspond to the integration point weights. Particle positions are initialized by looping over the set of elements, thus avoiding a space search algorithm for determining the bounding element for a particle. This particle initialization process however leads to particles having different volumes, since the particle volume in this approach is determined by the integration point weight. Moreover, the initial number of particles per element is somewhat limited. In [35], the initial number in a $\mathbb{R}^3$ tetrahedral element is for example limited to a maximum of 10 particles per element.

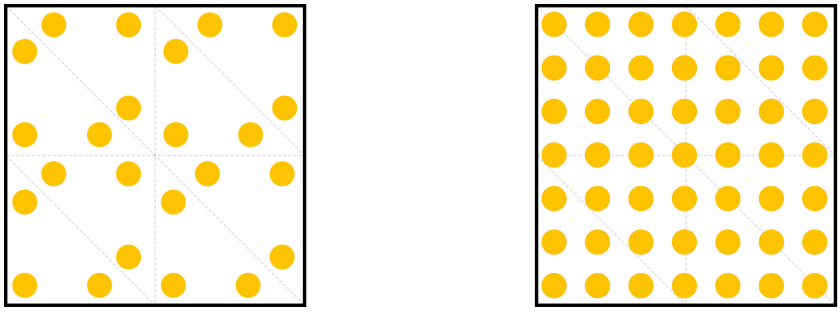


Figure 4.3: Integration point particle initialization (left panel) and particle initialization on regular lattice (right panel).

A different particle initialization procedure is therefore employed in this thesis. The idea is to place the particles at a regular Cartesian lattice with predefined spacing (i.e. the particle spacing), see Figure 4.3. This has the advantage of choosing the particle resolution arbitrarily of the background grid, although the method will in general lead to a mismatch between the total volume (and masses) of the particles in an element and the element volume (and mass).
The regular, rectangular lattice is generated in the bounding box of the geometric domain. In order to deal with domains of arbitrary, polygonal shape, particles outside the polygon are removed in a second step by performing a point-in-polygon test, see Section 4.5.1. Contrary to the particle initialization at the integration points, a space search algorithm is required for determining the bounding element for each particle.

4.2 Particle-to-grid mapping

With the chosen P_0P_1element in mind, an algorithm is presented to construct a velocity and density field at the background grid. For this, a least square approximation is used. Recalling the formulation of a least square projection (Eq. (3.21) and Eq. (3.22)) the weight functions are set to piecewise constants over an element and the basis functions are chosen to be piecewise continuous polynomials of degreee 0. With these choices, the monomial matrix $\mathbf{Q}$ and the weight function matrix $\mathbf{W}$ become, respectively:

$$\mathbf{Q} = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.1}$$

where, at element level, $\mathbf{Q}$ has size k and $\mathbf{W}$ has size $[k \times k]$ with $k = |P_e|$ and P_e defined as:

$$P_e = \{p \,:\, \mathbf{x}_p \in \Omega_e\} \tag{4.2}$$

Thus, matrix $\mathbf{A}$ (the moment matrix) simply becomes equal to the number of particles in the element:

$$\mathbf{A} = \mathbf{Q}^T\mathbf{I}\mathbf{Q} \tag{4.3}$$
$$= \mathbf{Q}^T\mathbf{Q} \tag{4.4}$$
$$= [k] \tag{4.5}$$

in which $\mathbf{I}$ the identity matrix. Furthermore, since the matrix $\mathbf{B}$ equals $\mathbf{Q}^T\mathbf{I}$, the least square map of quantity Ψ from the set of particles to element e becomes:

$$\begin{aligned} \Psi_e &= \mathbf{q}^T(\mathbf{x})\mathbf{A}^{-1}(\mathbf{x})\mathbf{B}(\mathbf{x})\Psi_p \\ &= \frac{1}{|P_e|}\sum_{p\in P_e}\Psi_p \end{aligned} \tag{4.6}$$

In other words, the element value (either the vector valued velocity, Eq. (3.50), or the scalar valued density, Eq. (3.51)) becomes the arithmetic mean of the particle properties. This least square mapping scheme

will be denoted as the 'least square mapping' in the sequel.
The grid representation of the state variables (i.e. mass and momentum), will in general not equal the particle representation, for the simple reason that generally:

$$\int_{\Omega_e} \frac{1}{|P_e|} \sum_{p \in P_e} \rho_p \mathrm{d}\Omega \neq \sum_{p \in P_e} m_p \tag{4.7}$$

And so it is observed that the total particle mass in an element does not equal the element mass.
Another issue with the density mapping will arise in the elements containing both air and water particles in two-phase simulations. Consider for example an element containing 3 air particles, having a density of $1\,\mathrm{kgm}^{-3}$ and 1 water particle, having a density of $1000\,\mathrm{kgm}^{-3}$. When using the arithmetic mean (i.e. Eq. (4.6)), the element density will be $\rho_c = \frac{1003}{4} \approx 251\,\mathrm{kgm}^{-3}$. However, if taking the harmonic mean in such an element, the density would equal $\rho_c = \frac{4}{3.001} \approx 1.333\,\mathrm{kgm}^{-3}$. So the arithmetic mean appears to work in favor of the heavy phase, whereas the harmonic mean works in favor of the light phase. The averaging type in two-phase elements is therefore far from trivial and has been a subject of several studies (e.g. [47], [56]). In the paper by Deubelbeiss and Kaus [56] it was concluded that an arithmetic mean gives the best results for density averaging, this can probably be explained from the fact that density gradients are smaller with this method compared to the harmonic mean. So although the arithmetic mean, resulting from the least square interpolation of particle properties, will be used throughout the thesis, it is good to keep the shortcomings for two-phase elements of this interpolation method in mind.
The last disadvantage of the above presented mapping to be mentioned, is that the number of particles in an element does not influence the density. So a cell with one particle having a density of, say, $1000\,\mathrm{kgm}^{-3}$ neighboring a cell with hundred particles of the same density of $1000\,\mathrm{kgm}^{-3}$ will have an equal density and hence an equal mass as the numerical integration is performed at the grid level using Gauss quadrature. It can therefore be anticipated that using only Eq. (4.6) can lead to regions which are overfilled or underfilled with particles, since there is no mechanism forcing particles to regions with a low particle concentration.
As an alternative, a different mapping approach is developed putting more emphasis on the particle distribution. For the chosen $\mathrm{P_0P_1}$element this is achieved as follows. Instead of doing the particle mapping with Eq. (4.6), the grid density field is obtained from:

$$\rho_e = \frac{1}{V_e} \sum_{p \in P_e} m_p \tag{4.8}$$

And the velocity mapping from particles to background grid is defined as:

$$\mathbf{u}_e = \frac{1}{\rho_e V_e} \sum_{p \in P_e} m_p \mathbf{u}_p \tag{4.9}$$

in which V_e the element volume.
For the $\mathrm{P_0P_1}$element these mappings can be regarded as an MPM approach for computing the integrals. In order to show this, consider for example the mass matrix for the $\mathrm{P_0P_1}$element. Using the denstiy mapping as presented in Eq. (4.8), the element mass matrix becomes:

$$\int_{\Omega_e} \rho N_i N_j \mathrm{d}\Omega = \int_{\Omega_e} \rho_e [1] \mathrm{d}\Omega \tag{4.10}$$

$$= \int_{\Omega_e} \frac{1}{V_e} \sum_{p \in P_e} m_p \mathrm{d}\Omega \tag{4.11}$$

$$= \frac{1}{V_e} \sum_{p \in P_e} m_p \int_{\Omega_e} \mathrm{d}\Omega \tag{4.12}$$

$$= \frac{1}{V_e} \sum_{p \in P_e} m_p V_e \tag{4.13}$$

$$= \sum_{p \in P_e} m_p \tag{4.14}$$

Which is indeed similar to the MPM approximation for integrals according to Eq. (3.4).
The above derivation shows that mass is locally conserved, that is, the element mass equals the particle mass. The same holds for the linear momentum when using Eq. (4.9). Therefore, this mapping type will be termed 'conservative mapping'.
The two above presented mapping approaches are compared in Section 5.1.6.

4.3 Implementation of the FEM discretization

4.3.1. Implementation of the weak form

In Section 3.2 the equations at the background grid were derived from the equations for momentum conservation Eq. (3.23) and mass conservation Eq. (3.24). In aforementioned section, first of all the spatial discretization using FEM techniques was performed after which time integration was conducted, resulting in the coupled system of equations Eq. (3.44). This method-of-lines strategy, in which the spatial discretization is performed first and finally the time integration is performed, is in fact only one approach for discretizing the equations.
Another popular approach for discretizing the incompressible Navier-Stokes or incompressible Euler equations are the fractional step methods. In this approach, the equations for the velocity and the pressure are solved separately and coupled by means of an intermediate velocity field. This 'physical splitting' [7] of the governing equations (Eqs. (3.23)-(3.24)) in fact means that time-integration is performed *before* the spatial integration. The idea behind the fractional step methods is that any vector field $\mathbf{w}$ can be decomposed in a solenoidal vector field $\mathbf{v}$ (i.e. $\nabla \cdot \mathbf{v} = 0$), the gradient of a scalar field ϕ which is by definition irrotational (since $\nabla \times (\nabla \phi) = 0$) and a harmonic field $\mathbf{h}$ which is both divergence free and curl free (i.e. $\nabla \cdot \mathbf{h} = \nabla \times \mathbf{h} = 0$). In other words, the vector field $\mathbf{w}$ can be written as:

$$\mathbf{w} = \mathbf{v} + \nabla \phi + \mathbf{h} \tag{4.15}$$

This principle is known as the Helmholtz decomposition and can be used to split the governing (time-discretized) equations in different steps, resulting in a series of relatively small systems to solve.
In this section it will be argued that the physical splitting of the governing continuum equations can be considered to be a special case of the algebraic splitting resulting from an incomplete LU-factorization of the system of equations Eq. (3.44). After showing this similarity between physical and algebraic splitting, the governing bilinear forms are recapped and the implementation of these equations in FEniCS is presented.
Following Quarteroni *et al.* [57], the system of equations Eq. (3.44) can be factorized as:

$$\begin{bmatrix} \Delta t^{-1}\mathbf{M} & \mathbf{0} \\ \mathbf{G}_k^T & -\Delta t \mathbf{G}_k^T \mathbf{M}^{-1} \mathbf{G}_k \end{bmatrix} \begin{bmatrix} \mathbf{U}_k^* \\ \mathbf{P}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_k^* \\ \mathbf{h}^{n+1} \end{bmatrix} \tag{4.16}$$

and

$$\begin{bmatrix} \mathbf{I} & \Delta t \mathbf{M}^{-1} \mathbf{G}_k \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{U}_k^{n+1} \\ \mathbf{P}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{U}_k^* \\ \mathbf{P}^{n+1} \end{bmatrix} \tag{4.17}$$

where the matrix and vector definitions can be found in Eqs. (3.40)-(3.43) and Eq. (3.45). Furthermore, $\mathbf{I}$ is the identity matrix.
This factorization results in the following sequence of operations:

$$\mathbf{M}\mathbf{U}_k^* = \Delta t \mathbf{b}_k^* \tag{4.18}$$

$$\Delta t \mathbf{G}_k^T \mathbf{M}^{-1} \mathbf{G}_k \mathbf{P}^{n+1} = \mathbf{G}_k^T \mathbf{U}_k^* - \mathbf{h}^{n+1} \tag{4.19}$$

$$\mathbf{M}\mathbf{U}_k^{n+1} = \mathbf{M}\mathbf{U}_k^* - \Delta t \mathbf{G}_k \mathbf{P}^{n+1} \tag{4.20}$$

Hence, the problem has been rewritten into three sequential steps, where it is noted that the final step is obtained by pre-multiplying with the matrix $\mathbf{M}$.
The matrix $\mathbf{G}_k^T \mathbf{M}^{-1} \mathbf{G}_k$ can be regarded to be the discrete version of the Laplace operator. It was shown in [57] that this operator is exactly similar to the discretization of the continuous Laplace operator under certain conditions. For a formal proof the reader is referred to the aforementioned paper, here it is only noted that this is indeed valid for the P_0P_1element (having an extremely simple mass matrix, with only diagonal entries).

Noting this, it is clear that the sequence of computational steps is similar to the sequence of steps following the fractional step method based on a physical splitting of the equations (see e.g. [7].) Thus, recalling the matrix and vector definitions, taking the external body force $\mathbf{f} = \rho\mathbf{g}$ with $\mathbf{g}$ the gravitational force and dividing all terms by ρ, the sequence of equations Eqs. (4.18)-(4.20) forms a representation of the following set of discrete bilinear forms.
Eq. (4.18): find $\mathbf{u}_h^* \in \mathcal{V}_h$ such that

$$\int_\Omega \mathbf{u}_h^* \cdot \mathbf{v}_h \mathrm{d}\Omega = \int_\Omega \mathbf{u}_h^n \cdot \mathbf{v}_h \mathrm{d}\Omega + \Delta t \int_\Omega \mathbf{g} \cdot \mathbf{v}_h \mathrm{d}\Omega \qquad \forall \mathbf{v}_h \in \mathcal{V}_h \tag{4.21}$$

The Pressure-Poisson equation, Eq. (4.19): find $p_h^{n+1} \in \mathcal{Q}_h$ such that:

$$\Delta t \int_\Omega \frac{1}{\rho^n} \nabla p_h^{n+1} \cdot \nabla q_h \mathrm{d}\Omega = \int_\Omega \mathbf{u}_h^* \cdot \nabla q_h \mathrm{d}\Omega - \oint_{\partial\Omega} \mathbf{u}_D^{n+1} \cdot \mathbf{n} q_h \mathrm{d}\Gamma \qquad \forall q_h \in \mathcal{Q}_h \tag{4.22}$$

And finally the end-of-timestep velocity, Eq. (4.20): find $\mathbf{u}_h^{n+1} \in \mathcal{V}_h$ such that:

$$\int_\Omega \mathbf{u}_h^{n+1} \cdot \mathbf{v}_h \mathrm{d}\Omega = \int_\Omega \mathbf{u}_*^{n+1} \cdot \mathbf{v}_h \mathrm{d}\Omega - \Delta t \int_\Omega \frac{1}{\rho^n} \nabla p_h^{n+1} \cdot \mathbf{v}_h \mathrm{d}\Omega \qquad \forall \mathbf{v}_h \in \mathcal{V}_h \tag{4.23}$$

Where it is emphasized that ρ_n and $\mathbf{u}_h^n$ follow from Eq. (4.6) for the least square mapping and from Eqs. (4.8)-(4.9) for the conservative mapping. The three variational formulations (Eqs. (4.21)-(4.23)) are easily implemented in FEniCS as:

```
# Define piecewise linear and piecewise constant function spaces
CG1s    = FunctionSpace(mesh,'Lagrange',1)
DG0v    = VectorFunctionSpace(mesh,'DG',0,dim=2)

u       = TrialFunction(DG0v)
v       = TestFunction(DG0v)
p       = TrialFunction(CG1s)
q       = TestFunction(CG1s)

ustar   = Function(DG0v)
uD      = Function(DG0v)
rhop    = Function(DG0v)

# Intermediate velocity
a0      =       inner(u,v)*dx
f0      =       inner(u0,v)*dx+dt*inner(fext,v)*dx

# Pressure Poisson equation, inhomogeneous Dirichlet at ds(1)
a1      =       dt*(inner(1/rhop*grad(p),grad(q)))*dx
f1      =       (inner(ustar,grad(q)))*dx - uD*q*ds(1)

# End-of-timestep velocity
a2      =       inner(u,v)*dx
f2      =       inner(ustar,v)*dx - dt*inner(1/rhop*grad(p1),v)*dx
```

4.3.2. Implementation of the viscous term

One of the drawbacks of the P_0P_1element mentioned in Section 4.1.1 is the non-trivial discretization of the viscous term appearing in the full Navier-Stokes equations due to the low order of the velocity space. Since the viscous term, for an incompressible fluid given by:

$$\boldsymbol{\sigma} = \mu\{\nabla\mathbf{u} + (\nabla\mathbf{u})^T\} \tag{4.24}$$

$$\nabla \cdot \boldsymbol{\sigma} = \nabla \cdot \mu\{\nabla\mathbf{u} + (\nabla\mathbf{u})^T\} \tag{4.25}$$

is of second order in the velocity, the derivation of the weak form and applying the standard Galerkin procedure would result in a weak form being first order in the velocity after integration by parts. Hence, using a standard procedure for the discretization of the viscous term would require at least piecewise linear basis functions whereas the P_0P_1element employs piecewise constant basis functions for the velocity.
Solution to this problem is to project the velocity gradient to a piecewise linear tensor field in order to approximate the (deviatoric) stress (Eq. (4.24)). Taking the weak form of Eq. (4.24) by multiplying with a tensor valued test function $\boldsymbol{\tau}_h \in \boldsymbol{\Gamma}_h$ and integrating over Ω results in:
Find $\boldsymbol{\sigma}_h \in \boldsymbol{\Gamma}_h$ such that:

$$\int_\Omega \boldsymbol{\sigma}_h : \boldsymbol{\tau}_h \mathrm{d}\Omega = \int_\Omega \mu \left[\nabla \mathbf{u}_h + (\nabla \mathbf{u}_h)^T\right] : \boldsymbol{\tau}_h \mathrm{d}\Omega \tag{4.26}$$

$$= \oint_{\partial\Omega} \mu \mathbf{u}_D \cdot \left[\boldsymbol{\tau}_h \mathbf{n} + \boldsymbol{\tau}_h^T \mathbf{n}\right] \mathrm{d}\Gamma - \int_\Omega \mu \mathbf{u}_h \cdot \left[\nabla \cdot \boldsymbol{\tau}_h + \nabla \cdot \boldsymbol{\tau}_h^T\right] \mathrm{d}\Omega \qquad \forall \boldsymbol{\tau} \in \boldsymbol{\Gamma}_h \tag{4.27}$$

where $\mathbf{u}_D$ follows from the Dirichlet boundary condition for the velocity and the tensor function space $\boldsymbol{\Gamma}_h$ is defined as:

$$\boldsymbol{\Gamma}_h = \{\boldsymbol{\tau}_h \in \mathbf{H}^1 : \boldsymbol{\tau}_h \in \mathbf{P}^k(\Omega_e) \ \forall e\} \tag{4.28}$$

with polynomial order $k \geq 1$.
A special remark should be made regarding the boundary integral in Eq. (4.27). As will be shown in the lid driven cavity test, this term can be used to enforce the no-slip condition in a weak sense. This seems somewhat counterintuitive, since the velocity boundary condition most often appears as an essential boundary condition in the model formulation for incompressible flows. However, it appears as a natural condition in the above presented formulation. It thus can be anticipated that the no-slip boundary condition can only approximately be enforced by means of Eq. (4.27). The accuracy of this implementation is assessed in Section 5.2 for the lid driven cavity problem.
By performing this projection, the divergence of the deviatoric stress tensor (Eq. (4.25)) can be evaluated without any problems, thus taking into account the shear stresses due to velocity gradients. In order to do so, the intermediate velocity step Eq. (4.21) is modified to (noting that the terms are divided by the density):
Find $\mathbf{u}_h^* \in \boldsymbol{\mathcal{V}}_h$ such that

$$\int_\Omega \mathbf{u}_h^* \cdot \mathbf{v}_h \mathrm{d}\Omega = \int_\Omega \mathbf{u}_h^n \cdot \mathbf{v}_h \mathrm{d}\Omega + \Delta t \int_\Omega \mathbf{g} \cdot \mathbf{v}_h \mathrm{d}\Omega + \Delta t \int_\Omega \frac{1}{\rho} (\nabla \cdot \boldsymbol{\sigma}_h^n) \cdot \mathbf{v}_h \mathrm{d}\Omega \qquad \forall \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h \tag{4.29}$$

With kinematic viscosity equaling $\nu = \frac{\mu}{\rho}$, the viscous term can be implemented in FEniCS as:

```
# Define function spaces (DG vector and CG tensor function space)
DG0v    = VectorFunctionSpace(mesh,'DG',0,dim=2)
CG1t    = TensorFunctionSpace(mesh,'CG',1)
# Defne trial and test functions
u       = TrialFunction(DG0v)
v       = TestFunction(DG0v)
sigma   = TrialFunction(CG1t)
tau     = TestFunction(CG1t)
# Define UFL functions and constants
u0      = Function(DG0v)
sigmav  = Function(CG1tP)
n       = FacetNormal(mesh)
fext    = Constant(g), nu      = Constant(1E-3)
# Variational form for deviatoric stress tensor
a0      =       inner(sigma,tau)*dx
f0      =       nu*( dot(ubc,dot(tau,n)) + dot(ubc,dot(tau.T,n)) )*ds
                - nu*( dot(u0,div(tau)) + dot(u0,div(tau.T)) )*dx
# Variational form for intermediate velocity
a1      =       inner(u,v)*dx
f1      =       inner(u0,v)*dx+dt*inner(fext,v)*dx+dt*inner(div(sigmav),v)*dx
```

4.3.3. Boundary conditions at mesh level

In Section 2.1.2 different boundary conditions were mentioned. This section briefly deals with the implementation of these boundary conditions at the mesh level. Section 4.5.3 deals with the issue how the different boundary conditions influence the particle updating process.

Periodic boundaries

Periodic boundary conditions are the simplest boundary conditions possible, in fact, this condition is not a true boundary condition as it represents an infinite extension of the domain. At the discrete level, the periodic boundary should be understood as a merging of the boundary nodes at the periodic boundaries. In FEniCS , the implementation of a periodic boundary condition is achieved by mapping the boundary nodes at the corresponding periodic boundary domain onto each other and constraining the function space definition. Details of implementing this in FEniCS are given in Appendix B.

Normal velocity boundaries

Recalling the weak form of the Pressure-Poisson equation (Eq. (4.22)), it is clear that the boundary term given by

$$-\oint_{\Gamma_D} \mathbf{u}_D^{n+1} \cdot \mathbf{n} q \mathrm{d}\Gamma \tag{4.30}$$

can be used to enforce the normal velocity, with Γ_D the part of the domain on which a Dirichlet condition for the velocity is specified. In case of a closed, fixed wall, the no-net normal flow condition is enforced by simply setting this term to zero.
Anticipating on the objective of setting-up a numerical wave flume, it is this term which can be used to implement a wave generating boundary condition. Recalling the hybrid particle-mesh nature of the method, implementation of the boundary term on a static background mesh would however lead to empty elements near the boundary in hybrid methods, Figure 4.4 .

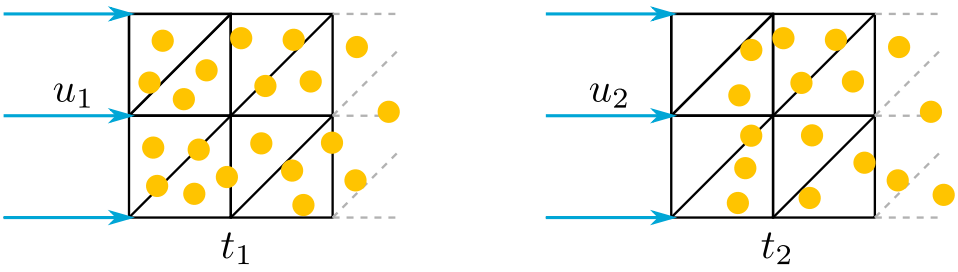


Figure 4.4: Kinematic boundary conditions imposed on a static background mesh can result in empty elements.

In the MPM community for solid mechanics, this problem has been solved by using a moving mesh approach in order to deal with non-homogeneous kinematic boundaries [35]. The idea is to keep the mesh boundary aligned with the prescribed boundary velocity, that is:

$$\mathbf{x}_b(t) \cdot \mathbf{n} = \int_{t^0}^{t} \mathbf{u}_D(\tilde{t}) \cdot \mathbf{n} \mathrm{d}\tilde{t} \tag{4.31}$$

in which $\mathbf{u}_D(t) \cdot \mathbf{n}$ the prescribed normal velocity at the boundary and $\mathbf{x}_b$ the location of the boundary.
The FEniCS implementation of such a moving mesh approach can be found in Appendix B. Important to note is that the system of equations has to be assembled every timestep, as the element volumes will change due to the grid movement.

Tangential velocity boundaries

Similar to the boundary condition for the normal flow, the tangential velocity can only be enforced in a weak sense, by specifying the boundary velocity $\mathbf{u}_D$ in the boundary term appearing in Eq. (4.27):

$$\oint_{\Gamma_D} \mu \mathbf{u}_D \cdot \left[\boldsymbol{\tau}_h \mathbf{n} + \boldsymbol{\tau}_h^T \mathbf{n} \right] \mathrm{d}\Gamma$$

In this thesis, the viscous term is only activated for problems having closed boundaries, therefore only an inhomogeneous tangential velocity component at the boundary will be considered (see Section 5.2). Furthermore, it is noted that setting $\mathbf{u}_D$ to zero, the no-slip condition is enforced in a weak sense at the boundary.

Open boundaries

Implementation of an open boundary condition is unavoidable when an inhomogeneous normal velocity is defined at the boundary in combination with a moving mesh. In such a case, the volume of the domain changes due to the boundary movement. Incompressibility of the material requires that either the domain does deform at other locations in order to keep a constant domain volume, or the fluid leaves the domain (i.e. an outflow, or open boundary condition). For the first approach, the compatibility condition

$$\oint_{\partial\Omega} \mathbf{u} \cdot \mathbf{n} \mathrm{d}\Gamma = 0 \tag{4.32}$$

has to be satisfied at the boundary. This is visualized in Figure 4.5, where u_1 and u_2 are defined such that the volume of the domain remains constant.

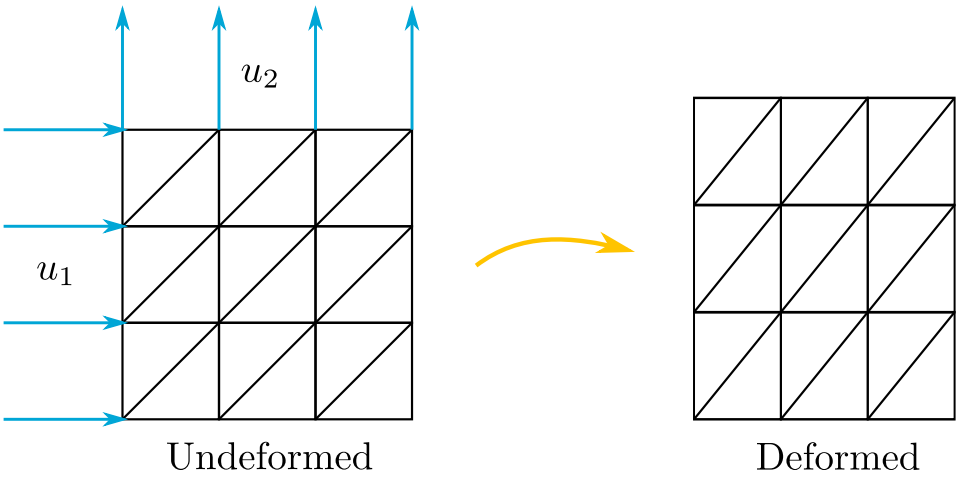


Figure 4.5: Imposing the boundary condition by a prescribed velocity.

At first glance, such an approach would have the benefit of avoiding open boundaries and, translated to a hybrid particle-mesh method, the amount of particles (that is, the total volume represented by particles) does not change in the computational domain. However, imposing this boundary condition will also implicitly assume the velocity to be uniform along the moving boundary, which is physically not correct. Implementation of an open boundary is therefore inevitable when applying the moving mesh approach.
Implementation of open boundaries falls apart in a grid related and a particle related part. The latter will be discussed in Section 4.5.3. The grid related part can be achieved by imposing a Dirichlet condition for the pressure along the open boundary, that is:

$$p(\mathbf{x}, t) = p_o(\mathbf{x}, t) \qquad \text{at} \quad \Gamma_o \tag{4.33}$$

with $\Gamma_o \subset \partial\Omega$ denoting the open boundary.
The FEniCS implementation of this boundary condition is presented in Appendix B.

4.4 Particle update

4.4.1. Grid-to-particle mapping

Solving the set of discrete equations results in a background grid pressure and velocity field at the new timestep. Particle properties needs to be updated based on the grid solution. However, at this point the drawback of the P_0P_1element is encountered because it solves for a piecewise constant velocity field, see Section 4.1.1. In FEniCS a piecewise linear, continuous velocity field is easily constructed from the

piecewise constant values using the project function:

```
# Project piecewise constants to piecewise linears
u1          =       project(u0,VectorFunctionSpace(mesh,'Lagrange',1,dim=2))
```

After obtaining the vertex velocities, while keeping the cell-center velocities (according to Figure 4.1), the velocity carried by the particle can be updated. For this, a traditional FLIP approach is used, so the particle velocities are updated based on a *velocity increment*. However, it is known that a pure FLIP velocity update can develop noise at particle level. Reason for this is the smaller amount of grid dofs compared to the particle dofs thus allowing subgrid velocity fluctuations to occur ([24],[32]). In order to oppress these high frequency velocity fluctuations, the FLIP update is blended with a small PIC fraction, in which particle velocities are overwritten by the grid velocity every timestep. The particle velocity is thus updated using Eq. (3.56):

$$\mathbf{u}_p^{n+1} = \alpha \sum_i N_i(\mathbf{x}_p^n)\mathbf{u}_i^{n+1} + (1-\alpha)\left\{\mathbf{u}_p^n + \sum_i N_i(\mathbf{x}_p^n)\left[\mathbf{u}_i^{n+1} - \mathbf{u}_i^n\right]\right\} \tag{4.34}$$

where the PIC fraction is denoted with α.

4.4.2. Particle advection

The final step of the algorithm is to advect the particles. As explained in Section 3.4, the grid velocity field is used for the particle advection. Thus the resulting ODE for the particle advection becomes:

$$\left.\frac{\mathrm{d}\mathbf{x}_p}{\mathrm{d}t}\right|_t = \tilde{\mathbf{u}}_p(\mathbf{x}_p, t) \tag{4.35}$$

where the velocity for the particle advection at time t is approximated as:

$$\tilde{\mathbf{u}}_p(\mathbf{x}_p, t) = \sum_i N_i(\mathbf{x}_p)\mathbf{u}_i(t) \tag{4.36}$$

The differece between the formulation for the *advected* velocity field (Eq. (4.34)) and the *advecting* velocity field (Eq. (4.36)) should be noted.
It can be anticipated that it is of utmost importance to use an accurate and preferably cheap scheme to approximate the solution Eq. (4.35). Accurate because the particle properties are propagated through the domain by means of this equation and cheap because the number of equations to solve is dependent on the amount of particles in the system, thus the advection step is relatively expensive compared to grid related steps. Therefore, a well-informed choice for an advection scheme should be made. To this end, three numerical schemes for solving Eq. (4.35) are investigated. In all these schemes the velocity at the particle position is obtained by mapping the grid velocities (Eq. (4.36)).

Euler scheme

Probably the most simple scheme for advecting the particle is the Euler scheme, or to be more precise the Euler-Cromer method, in which the particle position is simply updated as:

$$\mathbf{x}_p^{n+1} = \mathbf{x}_p^n + \tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^n)\mathbf{\Delta} t \tag{4.37}$$

Where it is noted that the velocity at the new time $n+1$, $\tilde{\mathbf{u}}_p^{n+1}$, is computed before the ODE (Eq. (4.35)) is solved. Furthermore, since the velocity $\tilde{\mathbf{u}}_p^{n+1}$ is evaluated at the old position $\mathbf{x}_p^n$, the resulting scheme is semi-implicit.
Although only first order accurate, the advantages of the Euler method are its simplicity and its low computational cost.

Heun scheme

A slightly more advanced method is the Heun scheme, in which not only the initial position $\mathbf{x}_p^n$ but also an intermediate particle position $\mathbf{x}_p^*$ is used to approximate the particle velocity. Thus, two steps are required:

a step to compute the intermediate postion, the so-called 'predictor' step, and a step to compute the final position, the so-called 'corrector' step:

$$\begin{aligned} \mathbf{x}_p^* &= \mathbf{x}_p^n + \tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^n)\Delta t \\ \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \frac{\Delta t}{2}\left\{\tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^n) + \tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^*)\right\} \end{aligned} \tag{4.38}$$

Obviously, the predictor step is the regular Euler-Cromer step, whereas the corrector step can be recognized as the trapezoidal method (in space). As a result of using a predictor and a corrector step, the Heun scheme takes the spatial variation of the velocity field into account, which is the main advantage over the Euler-Cromer method.

Runge-Kutta scheme

The concept of using one intermediate position as in the Heun scheme, can be extended to using two or more intermediate positions. The family of numerical schemes thus arising are the Runge-Kutta schemes. For the advection of particles the following three stage method (RK3 scheme) is recommended by Bridson [32], where the coefficients in the scheme were derived by Ralston [58] based on a minimization of the resulting error.

$$\begin{aligned} \mathbf{k}_{1,p} &= \tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^n) \\ \mathbf{k}_{2,p} &= \tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^n + 0.5\Delta t\mathbf{k}_{1,p}) \\ \mathbf{k}_{3,p} &= \tilde{\mathbf{u}}_p^{n+1}(\mathbf{x}_p^n + 0.75\Delta t\mathbf{k}_{2,p}) \\ \mathbf{x}_p^{n+1} &= \mathbf{x}_p^n + \frac{2}{9}\Delta t\mathbf{k}_{1,p} + \frac{3}{9}\Delta t\mathbf{k}_{2,p} + \frac{4}{9}\Delta t\mathbf{k}_{3,p} \end{aligned} \tag{4.39}$$

It must be emphasized that the time dependency of the velocity field is neglected in this scheme as well as in the previously presented form of the Heun scheme, thus the accuracy of the method can be expected to be first order in time [32].

Time step requirements

Since the above presented time integration schemes are explicit, it can be expected that the maximum allowable time step size is determined by the CFL-condition, stating that the ratio of the incremental propagation distance of information in the computational domain over the distance between grid nodes must be smaller than a certain threshold, in other words:

$$\mathrm{CFL} = \frac{U\Delta t}{h} \leq \epsilon \tag{4.40}$$

in which U an estimate for the maximum velocity, h a characteristic length scale for the element and the threshold value ϵ for the Euler forward scheme equals 1.
Although h is uniquely defined for regular, rectangular grids, its definition is ambiguous for triangular (or tetrahedral) meshes. As an estimate, the circumradius can be used as a measure for h, Figure 4.6 [59].

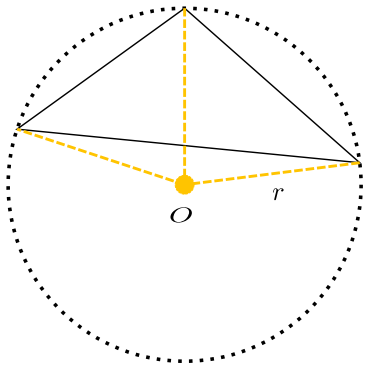


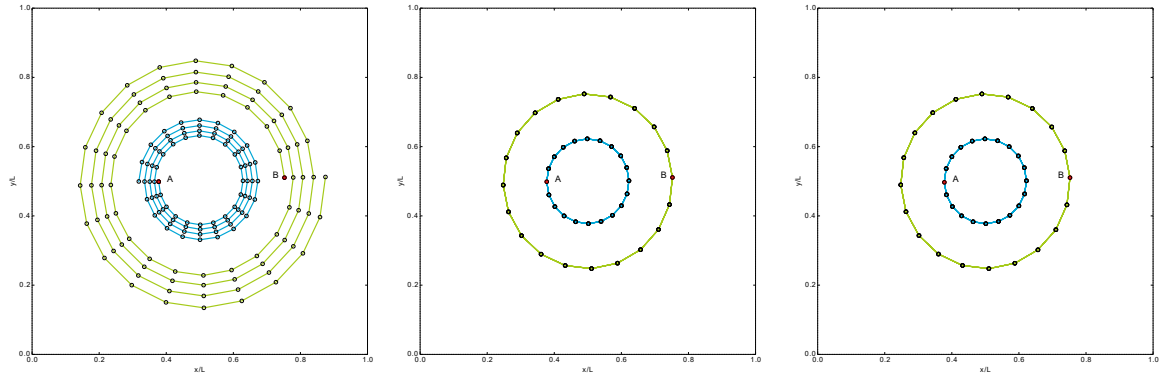
Figure 4.6: Circumradius r of a triangular element.

Test case: solid body rotation

As a test case for the advection schemes, the velocity field corresponding to a solid body rotation is imposed on the background grid. This (stationary) velocity field is given by:

$$\mathbf{u} = \begin{bmatrix} -\omega(y - y_0) \\ \omega(x - x_0) \end{bmatrix} \tag{4.41}$$

Given this velocity field, each particle path should describe a concentric circle around y_0 with a period of 2 s. Figure 4.7 presents the trajectories of particle A (with initial position (0.3775,0.5025)) and B (with initial position (0.7525,0.5025)) for the three different advection schemes. A 25x25 regular triangular background grid was used on the unit square computational domain ($\Omega = [0, 1]$x$[0, 1]$), timestep was 0.01 s, corresponding to a CFL-number of approximately 1 using Eq. (4.40).



(a) (b) (c)

Figure 4.7: Particle trajectories for point A and B in a solid body rotation for (a) the Euler scheme, (b) the Heun scheme and (c) the RK3 scheme. Red dots indicate initial positions of A and B.

The Euler scheme shows an inaccurate result as the particles are drifting away from the center of rotation, an observation which is confirmed in literature (see e.g. [32], [60]). The expected circular particle trajectories are obtained with both the Heun scheme and the RK3 scheme, revealing that at least one of these schemes is required for an accurate particle advection. Differences between the Heun scheme and the RK scheme are assessed by means of a simple convergence test varying the temporal resolution. Varying the spatial resolution to assess spatial convergence does not make any sense for the given test case, since the used barycentric interpolation is capable of representing linear functions exactly and so the linearly varying velocity field defined by Eq. (4.41) is exactly represented on the grid, irrespective of the grid resolution. The numerical error is expressed in terms of the position error as:

$$\epsilon = \frac{\|\mathbf{x}_p(t^1) - \mathbf{x}_p(t^0)\|}{\|\mathbf{x}_p(t^0)\|} \tag{4.42}$$

Taking $t^0 = 0$ s and $t^1 = 4$ s, the error is evaluated after two cycles. These errors are plotted in Figure 4.8 for four different timesteps, the right panel in this figure shows the computational time T for the different configurations. From this figure it becomes clear that the Heun scheme has a quadratic convergence rate in time, whereas the RK3 scheme appears to show a cubic convergence rate. In the left panel it is observed that although the error made with the RK3 scheme is several orders lower compared to the error made with the Heun scheme, the computational costs are of the same order. Based on the improved convergence properties combined with the low additional costs of the RK3 scheme compared to the Heun scheme, it is concluded to use the RK3 scheme for the particle advection.

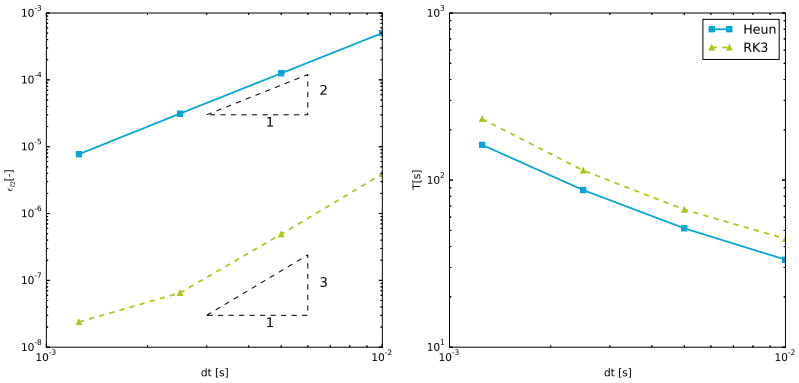


Figure 4.8: Postion error as defined in Eq. (4.42) (left panel) and computational times (right panel) as a function of the model time step compared for the Heun scheme and RK3 scheme. Position error is evaluated after two cycles.

4.5 Particle Administration

As discussed above, the quantities at the background grid are sampled from the particles. As the particles are freely moving through the (unstructured) background grid, every timestep a check has to be performed in which element the particle lives. This check basically falls apart in two substeps:

- Check if the particle lives in a given element.
- If this check fails, search for a neighboring element.

The first step is done by using a Point-in-Triangle test, the second step is done by means of a neighbor element searching algorithm. These two steps are described in Section 4.5.1 and Section 4.5.2. Moreover, depending on the boundary condition, particles crossing the boundary or close to the boundary need a special treatment. Influence of the specific boundary condition on the particle administration is described in Section 4.5.3.

4.5.1. Point-in-Triangle test

There are many strategies to determine whether a point is within a triangle, or more general, whether a point is in a polygon. One of the most intuitive methods, although hard to proof, is based on the Jordan curve theorem. This theorem states that if a point is within a polygon, any semi-infinite line drawn from that point intersects the boundary of the polygon an odd number of times, whereas an even number of intersections indicates that the point is outside the polygon, Figure 4.9. To perform the point-in-triangle test based on the Jordan curve theorem, an existing piece of freely available FORTRAN code was used [1].

4.5.2. Neighbor element searching

Not passing the point-in-triangle test means that the particle does not live in the given element, and so an algorithm needs to be constructed in order to search the updated element. The easiest method to achieve this would be to loop over elements and perform the point-in-triangle test for each element. However, such an algorithm is extremely expensive as it can result in performing the point-in-triangle test $n_p \times n_e$ times every timestep. Moreover, by using this approach the fact that the new position of a particle is most likely in the neighborhood of the old position is completely ignored. This notion is the key in constructing an efficient searching algorithm based on the barycentric technique. The idea is simple: imagine having the triangle with vertices at position $\mathbf{x}_0$, $\mathbf{x}_1$, $\mathbf{x}_2$ and the particle at position $\mathbf{x}_p$, further denote the vectors $\mathbf{p} = \mathbf{x}_p - \mathbf{x}_0$, $\mathbf{q} = \mathbf{x}_1 - \mathbf{x}_0$ and $\mathbf{r} = \mathbf{x}_2 - \mathbf{x}_0$, then it readily follows that a point $\mathbf{x}_p$ is in the triangle if and

[1] See: http://www.ecse.rpi.edu/~wrf/Research/Short_Notes/pnpoly.html#C%20Semantics

only if:

$$\begin{aligned} \mathbf{x}_p &= \mathbf{x}_0 + u\mathbf{q} + v\mathbf{r} \qquad \text{such that} \\ u + v &\leq 1 \\ u &\geq 0 \\ v &\geq 0 \end{aligned} \tag{4.43}$$

This equation is not only useful in determining whether a point is in a triangle[2] but also to determine through which element facet a particle escaped if the conditions in Eq. (4.43) are not satisfied. This is depicted in Figure 4.10. Knowing this facet, finding the neighboring element is straightforward and the point-in-triangle test (and the neighbor element searching algorithm if necessary) are repeated.

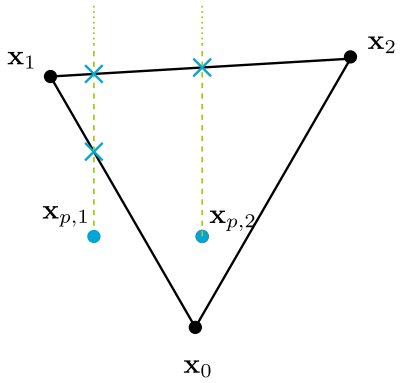


Figure 4.9: Principle sketch of Jordan curve theorem.

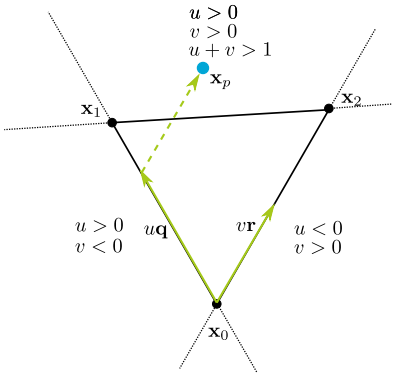


Figure 4.10: Definition sketch neighbor element search.

4.5.3. Boundary treatment of particles

Apart from enforcing the boundary conditions at the background mesh as described in Section 4.3.3, modifications at particle level are also required in order to incorporate the boundary conditions. The different 'particle treatments' required for the different boundary conditions are presented below.

Periodic boundaries

For proper implementation of the periodic boundaries at particle level a different element connectivity scheme is used in order to connect the elements at the opposing periodic boundaries. The modified element connectivity scheme has to ensure that the elements at opposing boundaries are connected, so that particles can, for example, freely leave the domain at the left and enter the domain at the right as if there is no boundary at all. This process is sketched in Figure 4.11. Apart from the modified connectivity scheme, no additional steps are required at particle level for these boundary conditions.

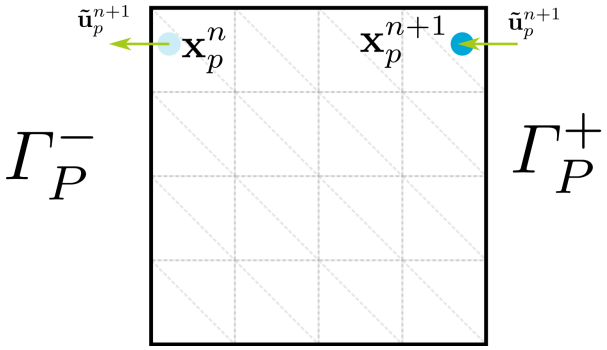


Figure 4.11: Periodic boundary condition at particle level.

[2] In fact, this approach was first used as a point-in-triangle test but it appeared to be prone to round-off errors compared to the method described in Paragraph 4.5.1

Normal velocity boundaries

By advecting the particles in the grid velocity field, the particles should not be able to penetrate the boundary when imposing a no-normal flow boundary condition (be it on a fixed mesh in case of a fixed solid wall, or on a moving mesh in case of a moving solid wall). Nevertheless, over the course of the computation, it has been observed that particles may slowly drift towards and eventually cross the domain boundary.

In order to prevent this unwanted crossing, an additional treatment of particles is required when particles are detected to cross the boundary. The idea is simple, initially a particle is assumed to be located at $\mathbf{x}_p^n$ and moved to position $\mathbf{x}_p^{n+1}$, where $\mathbf{x}_p^{n+1}$ denotes a particle position outside the domain. Say these points are connected by the vector $\mathbf{r} = \mathbf{x}_p^{n+1} - \mathbf{x}_p^n$. Since $\mathbf{x}_p^{n+1}$ is outside the domain, the position $\tilde{\mathbf{x}}$ where the particle p crosses the boundary becomes:

$$\tilde{\mathbf{x}} = \mathbf{x}_p^n + \beta \mathbf{r} \tag{4.44}$$

where $0 \leq \beta \leq 1$.

Following this approach, if a particle crossed the boundary, the new particle position is set to:

$$\mathbf{x}_p^{n+1} = \tilde{\mathbf{x}}^{n+1} \tag{4.45}$$

The particle velocity (i.e. the velocity *advected* by the particle) remains unaltered. This boundary-crossing procedure for the particle is sketched in Figure 4.12.

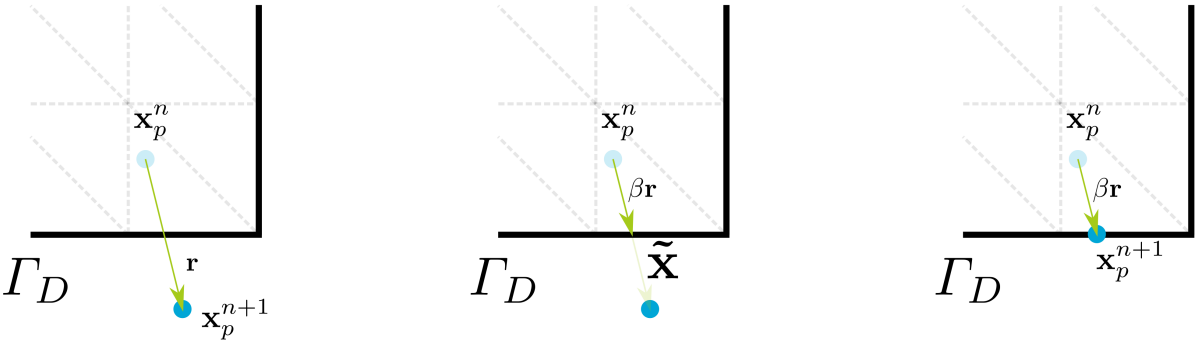


Figure 4.12: Boundary crossing procedure for particles near closed boundaries.

Although not considered in this thesis, the particle treatment at the normal velocity boundaries (or more generic at the velocity boundary) would become much more important when the particle velocities itself are used for the particle advection, see Section 3.4. In such a case, the particles freely move with their own velocity instead of a velocity dictated by the grid. Hence, boundary crossing of the particles can be expected to be much more pronounced. A popular boundary treatment in such a case is the bounce-back mechanism, where particles are reflected from the boundary (see e.g. [61] and [62]).

Tangential velocity boundaries

No modifications are required at particle level for the implementation of the tangential boundary condition.

Open boundaries

Although defining an outflow boundary on the grid is straightforward, the particle related part of this boundary condition is somewhat more troublesome. Outflow of particles is not that much of an issue, as particles are simply deactivated with a special flag when they cross the upper boundary, but the inflow of particles is much harder to deal with. The two issues related to the particle insertion are basically the questions: 1) when to insert particles and 2) where to insert particles.

As a simple solution to these questions, the following is proposed. Every timestep the volume of the elements having at least one vertex located at the open boundary, is compared to the volume of the particles in these elements. When the volume of the particles in such elements drops below a certain threshold, particles are inserted. So particles are inserted if:

$$\frac{\sum_{p \in P_{e_o}} V_p}{\int_{\Omega_{e_o}} \mathrm{d}\Omega} \leq 1 - \epsilon \tag{4.46}$$

with the summation running over the number of particles in boundary element e_o:

$$P_{e_o} = \{p \,:\, \mathbf{x}_p \in \Omega_{e_o}\} \tag{4.47}$$

In these formulas, e_o denotes the elements having at least one vertex located on the open boundary and ϵ denotes a threshold parameter, for which the exact value depends on the (approximate) number of particles per element. Typical values for this threshold parameter are in the order of 0.5.
Another question is where exactly to insert the particles. In the present implementation, this is done as follows. First, the amount of particles to be inserted in an element is determined, this amount of particles is added to the existing set of particles in the element. Next, all particle positions (including the existing particles) are scattered in the element using a uniform distribution and a pure PIC update is used to compute the new particle velocity (both for the existing and the new particles).
Although the two steps outlined above are quite rough, it can be argued that the motion of the water particles is of main interest, whereas the above described procedure will only influence air particles and thus can be expected to have a negligible or minor influence on the water motion.

Algorithm 2 Moving boundary approach for hybrid particle-mesh methods.

```
1:  Initialize grid.
2:  Initialize particles on regular lattice.
3:  while t < t_stop do
4:      Do Line 4 to Line 10 of Algorithm 4.
5:      for all elements with one vertex at the open boundary do
6:          if Eq.(4.46) then
7:              Add particles to element.
8:              Uniformly distribute particles over element.
9:              Overwrite particle velocity in element with PIC velocity.
10:     Move mesh and update velocity boundary.
11:     Update particle/mesh connectivity.
12: od
```

Implementing the above presented procedure in the code is only done in combination with inhomogeneous normal velocity boundary conditions at another part of the boundary. Together, the normal velocity and the open boundary form the 'moving boundary condition'. The implementation of this combination of boundary conditions is summarized in Algorithm 2.

4.5.4. Particle administration algorithm

Algortihm 3 summarizes the particle administration. In order to assess this algorithm a simple test is performed on an irregular triangular grid, containing 1 particle initially placed at position $(0.01, 0.01)$ in element e_0 and displaced to position $(0.99, 0.99)$ in a yet unknown element e_1. Using Algorithm 3 results in the element searching path presented in Figure 4.13, revealing that the algorithm returns a close to optimal path in order to find the new element.

Algorithm 3 Outline of particle administration algorithm for a single particle.

1: $e = e_0$
2: Point-in-Triangle test given $\mathbf{x}_p$ and e (Paragraph 4.5.1).
3: **while** $\mathbf{x}_p$ not in e **do**
4: $e_0 \leftarrow e$.
5: Find neighbor element e with barycentric technique (Paragraph 4.5.2).
6: Point-in-Triangle test given $\mathbf{x}_p$ and e (Paragraph 4.5.1).
7: **od**
8: $e_1 \leftarrow e_0$.

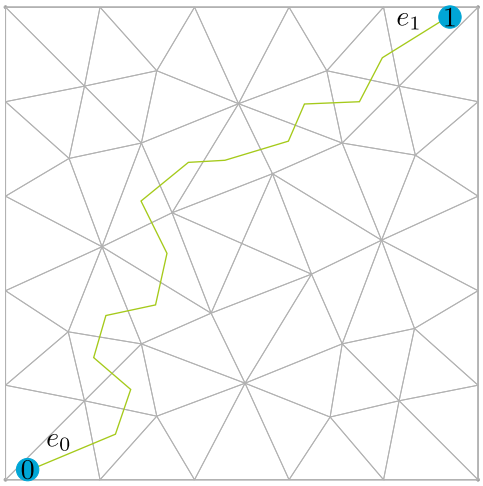


Figure 4.13: Performance of the element searching algorithm for a point displaced from position 0 to position 1.

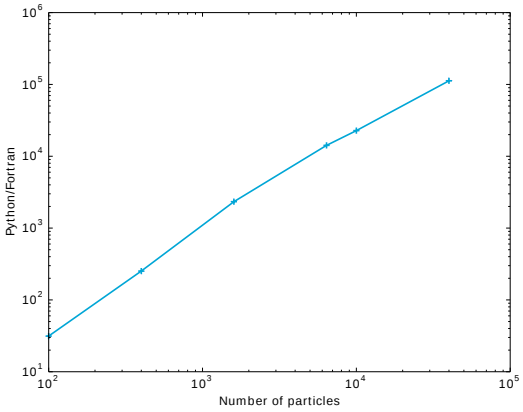


Figure 4.14: Ratio of the Python over FORTRAN computational time for performing the barycentric interpolation for different number of particles.

4.6 Implementation of the model

The above presented steps are implemented in a computer programme. Strategy followed is to use the scripting language Python as an interface only and use low level languages for the bulk of the work. Furthermore, as the background grid equations are discretized in a regular FEM way, the FEniCS finite element library is used (see Appendix A). Using existing code for the grid-related part an existing model infrastructure can be used. This has advantages in the pre-processing part (e.g. the grid generation), the processing part (the availability of dedicated matrix solvers) and the post-processing part (visualization).

Drawback of partially using existing code is that the existing software dictates more or less the possibilities. It is for example not possible to perform quadrature at particle positions in FEniCS , so a full MPM implementation can not even be tested in the current FEniCS version.
Furthermore, the FEniCS package does not offer tools to handle discrete particles. Therefore, the 'particle part' of the programme was coded in FORTRAN, interfaced by Python. In fact, for some parts of the particle update a combination of Python and FEniCS could be used instead of FORTRAN (e.g. evaluation of the basis functions at the particle locations). However, since this would involve (slow) Python loops, efficient and fast FORTRAN code is used for these particle related operations. As an example of the difference in performance between Python and Fortran, Figure 4.14 is presented. In this figure the ratio of the computational times for Python and FORTRAN required for the barycentric interpolation are compared for different numbers of particles. The difference in performance is clear. In fact, using Python for the particle related operations such as the barycentric interpolation quickly leads to excessively long computations.
Following this implementation strategy, a computational flowchart is presented in Figure 4.15.

4.7 Summary and conclusions

In this chapter the outline and implementation of the hybrid particle-mesh method was presented. Starting point were the incompressible Euler equations, which will be solved on a background grid employing the P_0P_1basis functions. At the beginning of each timestep a density and velocity field needs to be constructed from the particles. For this, a simple least square approximation is used, which for the specific implementation results in an arithmetic averaging procedure. As an alternative to this least square mapping, a mass and momentum conserving mapping was presented. Both mapping schemes will be compared in the next chapter.
After solving the governing equations at the background grid, a continuous velocity field is constructed from which the particle velocities are updated. Final step in the algorithm is to advect the particles in the continuous velocity field and keep track of the bounding element for each particle. It was shown that a higher order advection scheme is required. From a comparison of different numerical schemes it followed that the RK3 scheme is an economical choice.
The global outline presented in Algorithm 1 can thus be further refined, resulting in Algorithm 4 which can be regarded to be the basic formulation used in the remainder of the thesis.

Algorithm 4 Implicit particle method with P_0P_1element.

1: Initialize grid.
2: Initialize particles on regular lattice.
3: **while** $t < t_{\text{stop}}$ **do**
4: Construct piecewise constant velocity and density field with Eq. (4.6).
5: Compute intermediate velocity at grid with Eq.(4.21).
6: Compute pressure at grid with Eq. (4.22).
7: Compute final velocity at grid with Eq. (4.23).
8: Project piecewise constant velocities to a piecewise linear velocity field.
9: Update the particle velocity with Eq. (4.34).
10: Advect particles through grid velocity field with Eq. (4.39).
11: Do particle administration (Paragraph 4.5).
12: **od**

It is of utmost importance to be clear about the different assumptions made in this model implementation. Recognizing these steps will make clear where the code can be improved in future model development. These 'weak spots' in the model are identified to be the following:

- A simple least square implementation is used for the particle-to-grid mapping. Although this approach has the obvious advantage that elements with only air particles attain the true air density, and elements with only water particles attain the true water density, it does lead to a different representation of mass and momentum at the particle and at the grid level. Another disadvantage related

to this mapping procedure is that the particle cloud density does not play a role, and so there is no feedback to control the particle distribution.

- The element choice for this thesis is the P_0P_1element, which effectively avoids the locking problem. Drawback of this element choice is however the piecewise continuous nature of the veloctiy field. For the advection of the particles a continuous velocity field is required. Therefore, the piecewise constant, discontinuous velocities are mapped to a piecewise linear field. It is however noted that with this step, the resulting solution at the background grid (both used for updating the particle properties and positions) is not strictly divergence free anymore. Solutions to this issue can probably be found by considering alternative (admissible!) elements.

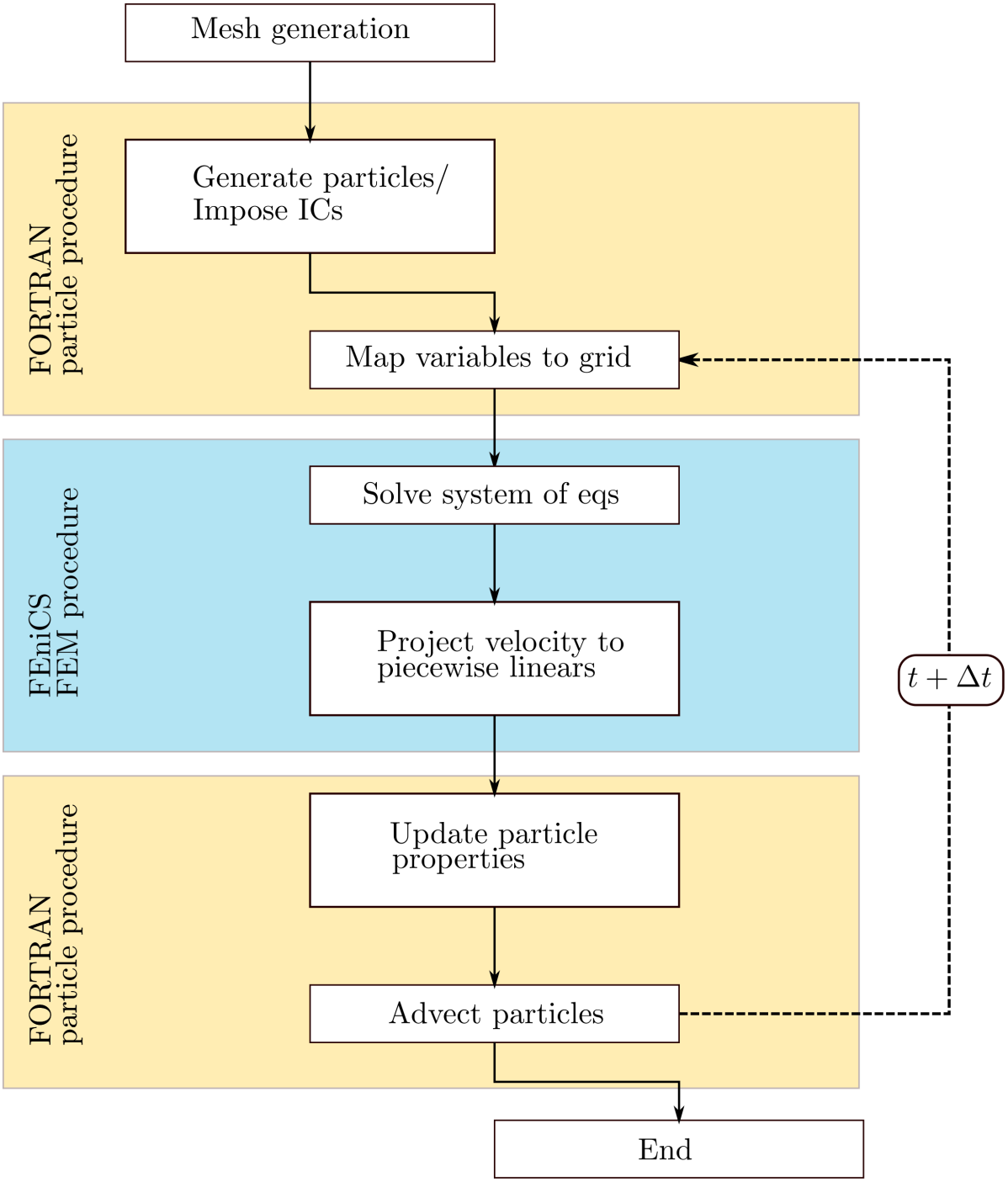


Figure 4.15: Flow chart of hybrid particle-mesh method using FORTRAN and FEniCS .

5

Testcases for single-phase flow

In Chapter 3 a generic outline for a hybrid particle-mesh method was formulated. Specific implementation of such a method was discussed in Chapter 4. In this chapter, the performance of the model implementation is assessed by running two well-known benchmark tests for single-phase flows: the Taylor-Green vortex problem and the lid driven cavity flow. Runnng these two test-cases will give a good insight in the temporal behavior of the scheme (Taylor-Green) and the ramifications of the specific particle-grid interaction implementation (lid driven cavity)[1].

5.1 Taylor-Green vortex problem

5.1.1. Problem description

The Taylor-Green vortex is used as a first case for testing the behavior of the scheme. This problem has closed analytical solutions for the velocity and the pressure fields, satisfying the incompressible Euler equations or the Navier-Stokes equations with periodic boundaries. For the Taylor-Green vortex, the velocity and pressure field are respectively given by:

$$\mathbf{u}(\mathbf{x}, t) = U e^{-2\nu\pi^2 t} \left[-\cos(k_x x)\sin(k_y y)\ ,\ \sin(k_x x)\cos(k_y y)\right]^{\mathsf{T}} \tag{5.1}$$

$$p(\mathbf{x}, t) = \frac{\rho}{4} e^{-4\nu\pi^2 t} \left(\cos(2k_x x) + \cos(2k_y y)\right) \tag{5.2}$$

with U the velocity amplitude, ν the kinematic viscosity and k_x and k_y the wavenumber in horizontal and vertical direction respectively. A plot of the analytical results on the unit-square domain $\Omega = [-1, 1]^2$ is given in Figure 5.1.

For the incompressible Euler equations, i.e. $\nu = 0$, the exponential decay term both present in the expression for the velocity and the pressure vanishes. Thus a stationary solution is expected when simulating the Taylor-Green vortex problem using the incompressible Euler equations, whereas an exponential decay is expected when using the full Navier-Stokes equations.
Given the closed analytical solutions, the testcase is a valuable tool in assessing temporal and spatial behavior of the numerical scheme, and provides a simple means of recognizing numerical diffusion. This section will therefore deal with different model configurations in order to get a better understanding of the numerical properties of the developed particle-mesh model. Outline is as follows: in Section 5.1.2 the basic model configuration is presented, in Section 5.1.3 the influence of the background grid is assessed, Section 5.1.4 discusses the influence of the PIC fraction, the temporal behavior for a viscous fluid is assessed in Section 5.1.5, Section 5.1.6 compares the least square particle-to-grid mapping with the conservative particle-to-grid mapping which were presented in Section 4.2.

[1] All computations in this thesis are run on a Dell Precision T5810 with Intel Xeon Quad Core E5-1620 CPU

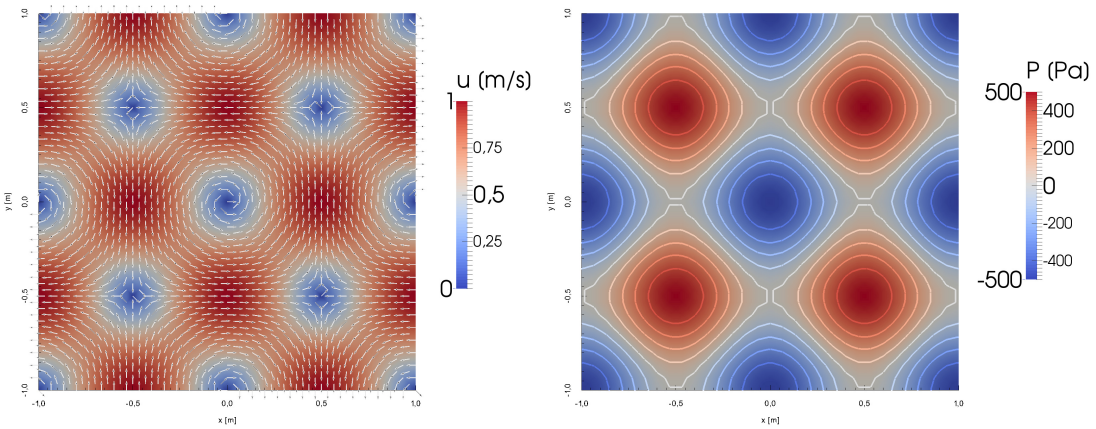


Figure 5.1: Exact velocities (left panel) and pressures (right panel) for the Taylor-Green problem on the domain $\Omega = [-1, 1]^2$ at $t = 0$ with $U = 1\,\mathrm{ms}^{-1}$ and $k_x = k_y = \pi\mathrm{m}^{-1}$.

5.1.2. Model configuration

Basic model settings are presented in Table 5.1. Given the grid properties and the velocity amplitude of $1\,\mathrm{ms}^{-1}$ the timestep of 0.05 s corresponds to a CFL number of approximately 0.1. The particles have an initial velocity obtained by evaluating Eq. (5.1) at $t = 0$, Figure 5.1. Furthermore it is noted that the problem does only contain periodic boundaries.

Table 5.1: Model properties Taylor-Green vortex problem.

Model parameters	Ω	$[-1, 1]^2$m	Computational domain
	nx×ny	30×30	Cells in x- and y-direction (crossed type)
	Δx_p	0.005 m	Particle spacing
	α	0.	PIC fraction
	U	$1\,\mathrm{ms}^{-1}$	Velocity amplitude
	Δt	$5E-3$ s	Time step
	ρ	$1000\,\mathrm{kgm}^{-3}$	Density
	ν	$0\,\mathrm{m}^2\mathrm{s}^{-1}$	Kinematic viscosity
	k_x, k_y	π	Wave length
	T	$\frac{D}{U}$	Dimensionless time
Computational aspects		160000	Number of particles
		44	Average nr. particles per element
		10 s	Simulated physical time
		16 min	Model run time

Pressure contours at $t = \Delta t$ and $t = 7.5$ s are presented in Figure 5.2, showing that the global pattern of the solution is well maintained over the simulation time. However, it also becomes immediately clear that the maximum and minimum pressures decrease over time whereas the pressure should be constant over time (Eq. (5.2)). This somewhat dissipative behavior of the scheme becomes better visible when comparing the exact and simulated velocities along the x- and the y-axis for different time instances. Figure 5.3 shows the dimensionless velocity components at $t = 2$ s, $t = 5$ s and $t = 7.5$ s. The simulated velocity profiles are in good agreement with the exact velocity profiles, although a slight velocity decay can be observed. As a result of this velocity decay due to numerical diffusion, the pressure extrema decrease. This decay is clearly visible in Figure 5.4, showing the dimensionless pressures at aforementioned time instances along the central vertical axis. At $t = 7.5$ s the minimum pressure is almost halved compared to the beginning of the simulation, and thus compared to the analytical solution.
So although exact and simulated results are in good agreement with each other, it is clear that the numerical scheme is not free of numerical diffusion. An artifact which is clearly visible for problems without external energy input such as the Taylor-Green vortex.

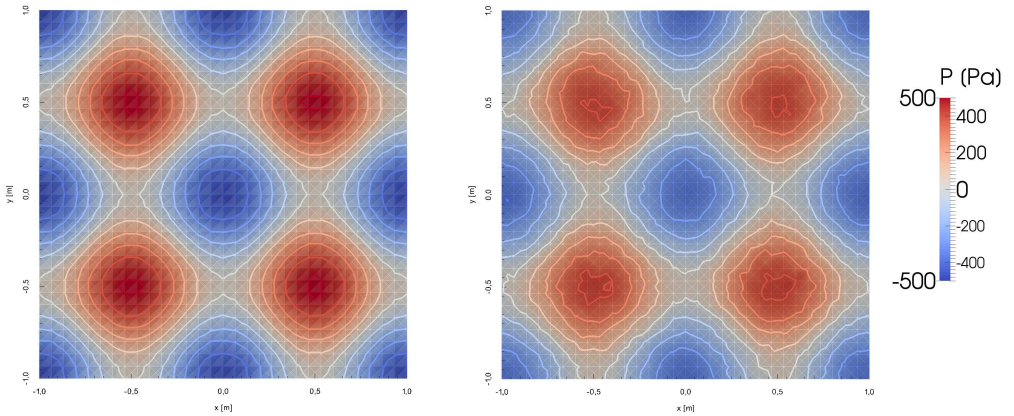


Figure 5.2: Pressure contours at $t = \Delta t$ (left panel) and $t = 7.5\,\mathrm{s}$ (right panel) on a regular triangular grid. Note the computational mesh overlay at the background.

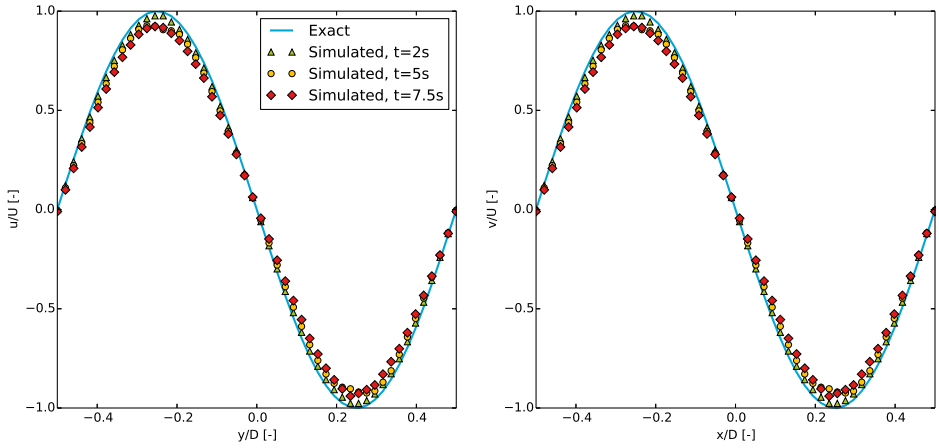


Figure 5.3: Normalized analytical and simulated velocity profiles in the Taylor-Green vortex at different time instances. Profile of horizontal velocity component at $x = 0.0$ (left panel), profile of vertical velocity component at $y = 0.0$ (right panel).

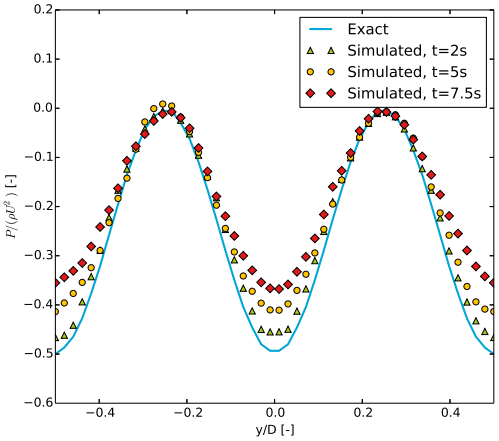


Figure 5.4: Normalized analytical and simulated pressure profiles in the Taylor-Green vortex at different time instances at $x = 0.0$.

5.1.3. Influence of the background grid

Grid resolution

As the governing equations are solved at a background mesh, the numerical solution obtained for various mesh resolutions is compared to the analytical solution. Six different regular triangular meshes are constructed with element circumradii ranging in between 0.4 m ((nx,ny)=(5,5)) and 0.04 m ((nx,ny)=(50,50)). Obviously, in hybrid particle-mesh methods, the grid size cannot be chosen independently of the particle spacing, since too fine a grid compared to the particle resolution may result in empty elements, thus rendering the computation instable or at least inaccurate. On the other hand, it is not clear how the amount of particles has to scale with the grid size. Therefore, an initial particle spacing $\Delta x_p = 0.0025\,\text{m}$ was used for all grid resolutions so that even for the most refined grid, the amount of particles in one element is about 130. All other parameters are kept similar to those presented in Table 5.1.
The accuracy of the different model runs is expressed in the L^2 errornorm of the velocity, evaluated at the background grid, i.e.:

$$\epsilon_{L^2} = \int_\Omega \|\mathbf{u}_h - \mathbf{u}_e\|^2 \mathrm{d}\Omega \tag{5.3}$$

with $\mathbf{u}_h$ the approximate solution and $\mathbf{u}_e$ representing the exact solution. FEniCS offers some useful tools when it comes to error analysis, so Eq. (5.3) is translated into UFL-code (see Appendix A) as:

```
# L2-norm velocity
M_u     =       inner((u_h-u_e),(u_h-u_e))*dx
u_err   =       assemble(M_u)
```

Results for the convergence test are presented in Figure 5.5. Interpretation of these results is not immediately straightforward, as no constant convergence rate is observed. In general it can be said that the convergence rate appears to decrease for finer grid meshes, negative convergence rates are observed for the refinement from a (nx,ny)=(40,40) grid to a (nx,ny)=(50,50) grid.

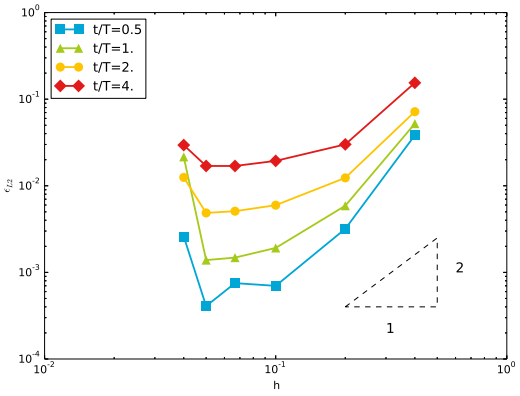


Figure 5.5: Velocity error for different grid spacings at various time instances, with $T = U/D$.

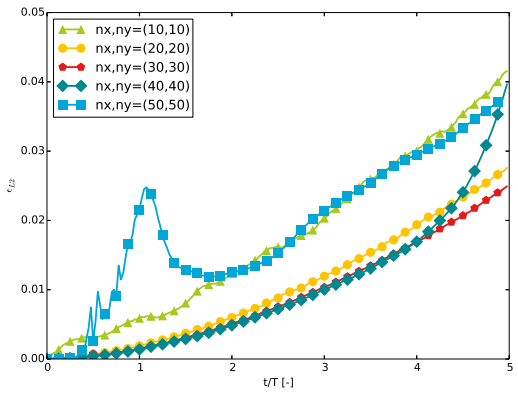


Figure 5.6: Velocity error over time for different grid resolutions. The nx,ny=(5,5) grid resolution is not shown for clarity reasons.

In order to explain the poor convergence behavior, the errornorm over time for the different meshes are plotted in Figure 5.6 showing an approximately expected behavior for all mesh sizes, except for the the (nx,ny)=(40,40) mesh and the (nx,ny)=(50,50) mesh. For the latter, the error quickly grows between $\frac{t}{T} \approx 0.5$ and $\frac{t}{T} \approx 1.5$ and decreases afterwards. Only explanation for this is that empty elements occur for the fine mesh, rendering the computation inaccurate. This apparently and unfortunately happens even for an initial particle resolution of approximately 130 particles per element. The reason of the error decay after $\frac{t}{T} \approx 1.5$ is that empty elements are given an extremely low, artificial density and hence a pressure gradient and a flow of particles towards the empty elements. It is however noted that, although such a mechanism does stabilize the algorithm, it does not fully resolve the underlying fundamental problem of violating the incompressibility constraint at the particle level. This observation therefore calls for an

additional constraint in order to control the particle distribution, an issue we already briefly touched upon in Section 4.2.
In summary, two observations are made based on the grid convergence test:

- Contrary to Eulerian methods, spatial convergence in hybrid mesh-particle methods is not just a matter of refining the (background) grid but also of using the optimal particle resolution. In the present example, a characteristic element size of 0.065 m (corresponding to the (nx,ny)=(30,30) grid) and a particle spacing of 0.0025 m appears to give the best result.
- Empty elements may lead to inaccurate model results. Reason for the occurrence of empty elements, is that the incompressibility constraint is enforced at grid level and so the particle distribution does not necessarily satisfy this constraint.

Grid regularity

Whereas a regular triangular mesh was used in the previous paragraph, the algorithm should work for unstructured meshes as well. In fact, only with an unstructured mesh the full benefits of FEM are exploited as it allows to vary the spatial resolution throughout the domain. Using unstructured meshes in FEniCS is a trivial task by importing meshes generated with well-established packages such as GMSH. So running the Taylor-Green vortex testcase for an unstructured triangular grid is rather a check to assess the performance of the hybrid scheme using the added particle libraries.

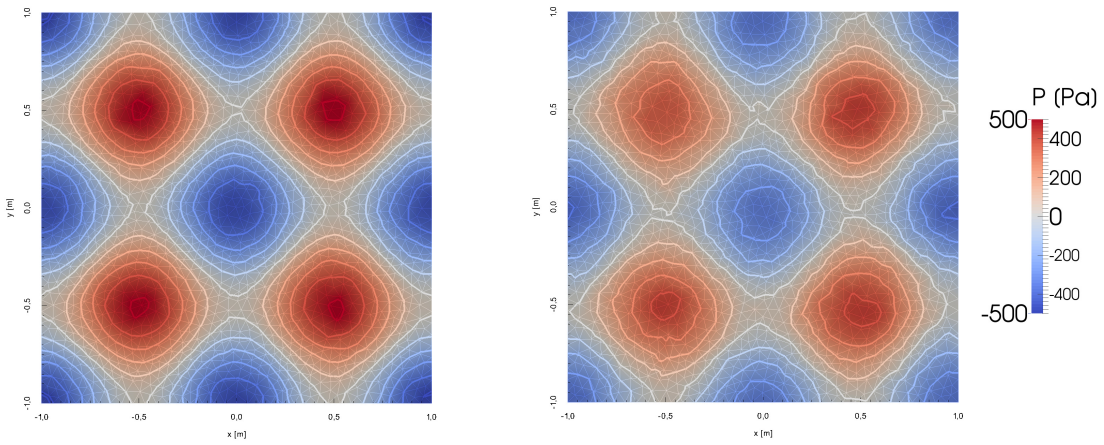


Figure 5.7: Pressure contours at $t = \Delta t$ (left panel) and $t = 7.5\,\mathrm{s}$ (right panel) on an irregular grid. Note the irregular computational mesh overlay at the background.

Figure 5.7 shows the solution on both the regular and the irregular triangular background grid. A value of 0.05 m was specified as a characteristic length scale of the elements. All other parameters were kept constant. As can be seen in this figure the solution is well-maintained on the irregular mesh. This is further confirmed by assessing the centerline velocities, Figure 5.8. The centerline velocities obtained with the irregular mesh are nearly identical to the exact solution and to the solution obtained on the regular mesh. In terms of pressures, the pattern is the same: the results on the irregular triangular grid are in good agreement with the exact solutions, but especially at later time instances the pressure trough in the center of the domain is underpredicted. A result wich has been observed in all Taylor-Green simulations seen so far (see e.g. Figure 5.4).
The test case thus confirms a correct implementation of the added particle libraries as well as the ability of the model to deal with fully unstructured meshes. The latter will turn out to be advantageous when dealing with irregular domains (see Section 6.4.2).

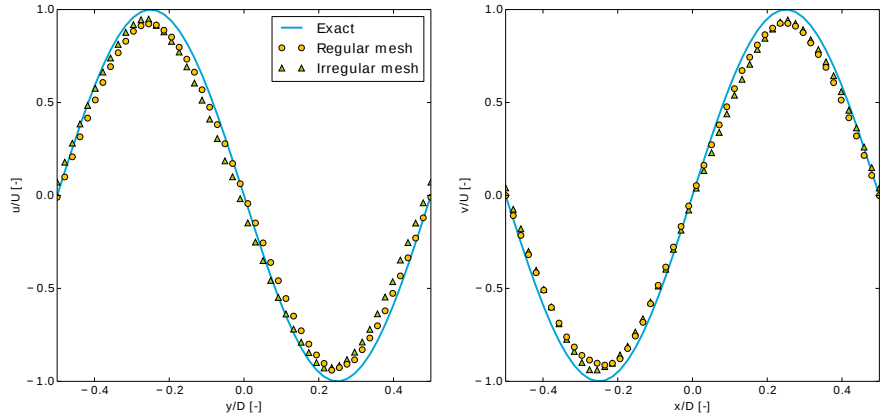


Figure 5.8: Normalized analytical and simulated velocity profiles in the Taylor-Green vortex at $t = 7.5\,\mathrm{s}$. Profile of horizontal velocity component at $x = 0.0$ (left panel), profile of vertical velocity component at $y = 0.0$ (right panel).

5.1.4. Influence of the PIC fraction

When discussing the grid-to-particle mapping in Section 4.4.1 a blended PIC-FLIP velocity update was presented. Based on literature it was argued that in a pure FLIP approach numerical diffusion is almost absent, but noise can develop at particle level in such an approach due to aliasing. A pure PIC update effectively avoids this by filtering the particle information every timestep with the background grid. Subgrid particle oscillations are effectively avoided with this method, but the approach was shown to be extremely diffusive.

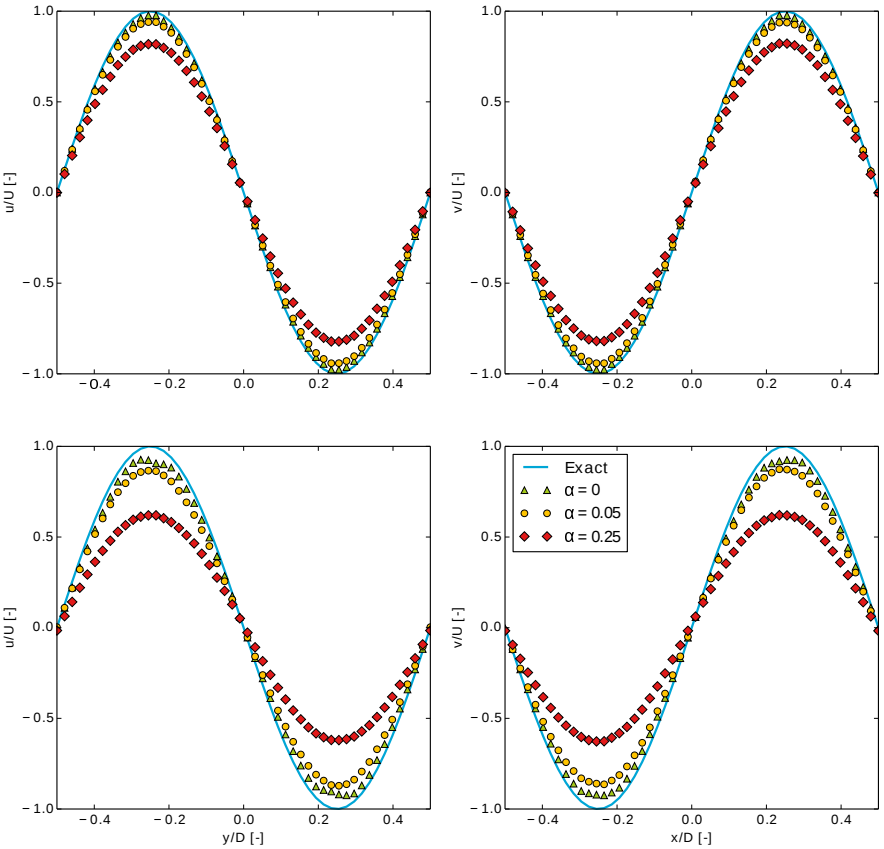


Figure 5.9: Normalized analytical and simulated velocity profiles in the Taylor-Green vortex at $t = 2\,\mathrm{s}$ (top row) and $t = 5\,\mathrm{s}$ (bottom row) for different PIC fractions α. Profile of horizontal velocity component at $x = 0.0$ (left column), profile of vertical velocity component at $y = 0.0$ (right column).

As a compromise to this controversy, the blended PIC-FLIP update is often used. In this approach the particle properties are updated using a weighted combination of a PIC update and a FLIP update (see Eq. (3.56)). For the Taylor-Green test case good results were obtained, even when using $\alpha = 0$ (i.e. a pure FLIP update). It is nevertheless useful to get a qualitative understanding of the role played by the blending parameter.
To this end, the simulated results obtained for the Taylor-Green test case are compared for three different PIC-fractions: 0, 0.05 and 0.25. All other parameters are kept similar to those presented in Table 5.1. The resulting centerline velocity profiles are presented in Figure 5.9 at time instances $t = 2\,\mathrm{s}$ and $t = 5\,\mathrm{s}$. The pattern is clear: whereas the velocity profile is only slightly damped in the absence of the PIC fraction ($\alpha = 0$) the velocity decay is much more pronounced for the higher PIC fractions. It is thus clear that the amount of numerical diffusion is significantly increased when using higher PIC fractions.
An interesting question is how large the numerical diffusivity in the scheme is. In other words: can the numerical diffusion be related somehow to a physical viscosity, or does it act similar to a physical viscosity? Altough a closed answer to this question would require an in-depth mathematical assessment of the numerics and would be too detailed for this thesis, the question can qualitatively be answered by considering the different PIC fractions and compare them to solutions for the Taylor-Green vortex problem for different viscosities.

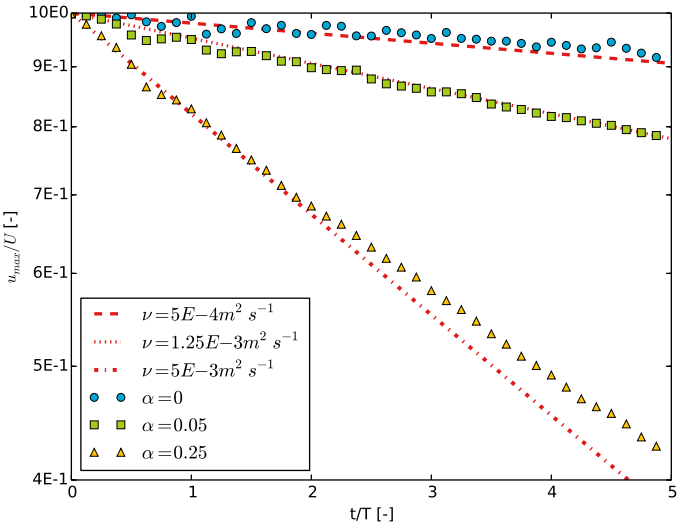


Figure 5.10: Velocity decay for different PIC fractions compared with analytical velocity decay for different kinematic viscosities.

Again, the Taylor-Green vortex problem is very useful in this, as analytical expressions are provided by Eq. (5.1) for the velocity. The maximum velocity amplitude and hence the maximum velocity in the domain is given by $U_{\mathrm{max}} = U_0 e^{-2\nu\pi^2 t}$. By comparing these maximum analytical velocities to the simulated maximum velocities, an estimate for the magnitude of the numerical diffusion can be given. Results for this are presented in Figure 5.10. A few things are observed in these figures:

- First of all, the simulated velocity decays are on approximately straight lines in the semi-log plot. This also holds for the velocity amplitude decay term for different viscosities. This qualitatively reveals that numerical diffusion acts similar as physical diffusion.
- Secondly, even for $\alpha = 0$ (so a pure FLIP update) the numerical diffusion is in the order of $1E-4\,\mathrm{m^2 s^{-1}}$ and thus an order 10^2 larger than the kinematic viscosity of water.

The numerical diffusion can be expected to be caused by the interpolation process of grid properties to the particles. It is therefore to be remarked that the numerical diffusion will be strongly dependent on the grid resolution, where lower numerical diffusions can be expected for finer meshes. This is also confirmed in literature, where for regular Cartesian meshes the relationship between the PIC fraction and the kinematic

viscosity is reported to be [32]:

$$\alpha = \frac{6\Delta t \nu}{\Delta x^2} \tag{5.4}$$

with Δx the grid spacing for regular Cartesian meshes.

5.1.5. Assessing the viscous term implementation

Including the viscous term, activates the exponential decay term in the analytical solutions for the Taylor-Green vortex problem. Model runs for different kinematic viscosities are compared against the analytical solution. All other parameters are similar to those defined in Table 5.1. In Figure 5.11 the simulated and analytical solutions are compared to the analytical solution for different viscosities at $t = 5\,\mathrm{s}$ along the central axes, showing a very good agreement between simulations and the theory. In terms of the pressures, the decay is too fast. A feature already observed in Figure 5.4. In order to get better insight in the velocity decay over time, the maximum simulated (grid) velocity is compared to the theoretical maximum velocity, given by $U_{\mathrm{max}} = U_0 e^{-2\nu\pi^2 t}$, Figure 5.13. As expected, the decay rate for the low viscosities $\nu = 2E-3\,\mathrm{m^2 s^{-1}}$ and $\nu = 2E-6\,\mathrm{m^2 s^{-1}}$ is slightly too high, which is a result of the numerical diffusion being larger or approximately of the same order as the physical diffusion. For the high viscosities, the numerical diffusion is negligible compared to the physical diffusion, decay rates are consequently captured well by the model. This reveals that the discretization of the viscous term presented in Section 4.3.2 gives the expected results, at least in the absence of any boundaries.

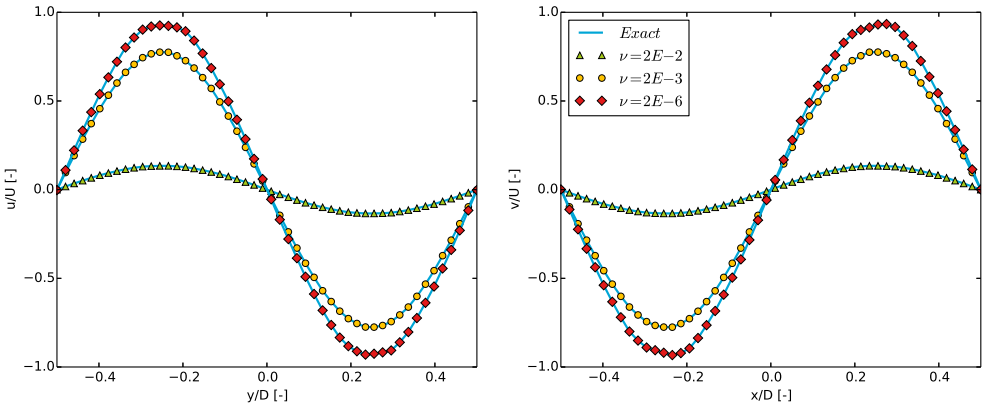


Figure 5.11: Normalized analytical and simulated velocity profiles in the Taylor-Green vortex at $t = 7.5\,\mathrm{s}$. Profile of horizontal velocity component at $x = 0.0$ (left column), profile of vertical velocity component at $y = 0.0$ (right column).

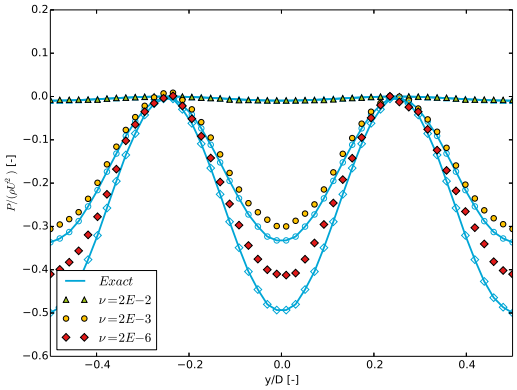


Figure 5.12: Normalized analytical and simulated pressure profiles for the Taylor-Green vortex at $t = 7.5\,\mathrm{s}$ and $x = 0.0$.

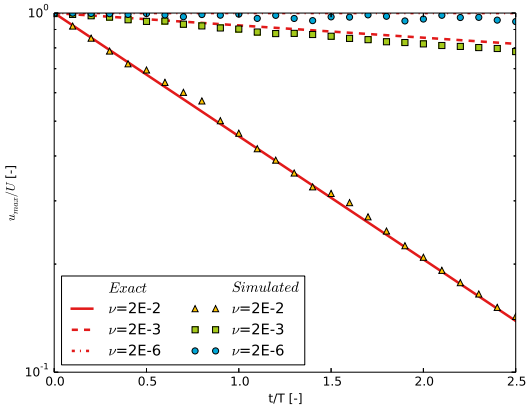


Figure 5.13: Velocity decay over time for different kinematic viscosities with $T = U/D$.

5.1.6. Particle distribution

During the grid convergence test for the fine grids the occurence of empty elements render the computation to be extremely inaccurate. It was found that even for an initial distribution of approximately 130 particles per element, this phenomenon may occur which may be attributed to the fact that there is too little control over the particle distribution during the computation. This is a known issue both in fully Lagrangian methods (see e.g. [14]) and hybrid particle-mesh methods (see e.g. [15], [18] and [45]). It can however expected to be especially pronounced for the approach used in this thesis. As mentioned in Section 4.2 this can be attributed to the fact that the number of particles in an element does not influence the cell density and so the derived approach lacks any control over the particle distribution.
Different approaches can be employed in order to mitigate ths problem. A first one is to add a particle distribution constraint to the Pressure-Poisson equation. The idea of such a constraint is to keep the 'nodal particle distribution' constant during the computation, see Figure 5.14. This means that the particle distribution should stay constant from the perspective of the background grid. Such a constraint can be cast in the following expression:

$$N_i(\mathbf{x}_p(t)) = N_i(\mathbf{x}_p(t_0)) \tag{5.5}$$

where N_i is the basis function corresponding to node i and $\mathbf{x}_p$ the position of particle p.

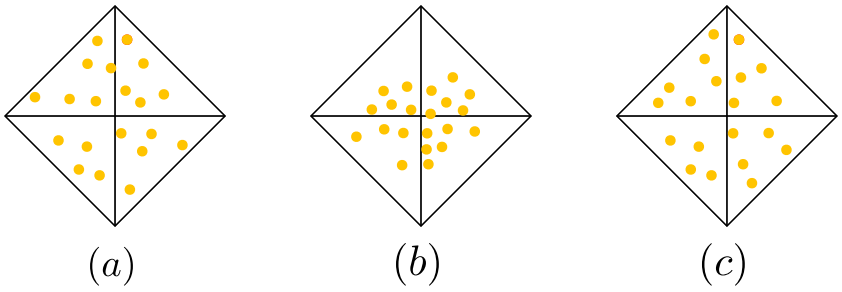


Figure 5.14: Initial particle distribution (a), particle distribution after perturbation (b) and particle distribution after imposing constraint Eq. (5.5) (c).

The above presented constraint has been implemented for a pure kinematic test case, resulting in a Poisson equation which was solved in an iterative way in order to update the particle positions. Although converging slowly, the iterative scheme resulted in the desired particle distribution for the pure kinematic test case. However, attempts to extend the 'nodal particle distribution constraint' to dynamic cases failed. Only a minor improvement of the particle distribution was observed whereas the amount of additional work by iteratively solving the Pressure-Poisson equation and iteratively advecting the particles required a

significant amount of additional compuational work. For the time being, this research path is therefore not further investigated.
Therefore, in this section only a comparison will be made between the least square mapping Eq. (4.6) and the conservative mapping Eqs. (4.8)-(4.9). The resulting particle distribution for both approaches are qualitatively compared for the Taylor-Green vortex problem at $t = 3\,\mathrm{s}$. It can be expected that irregularities in the particle distribution are best observed near stagnation points where the flow separates. Therefore, only the part of the domain around the lower left stagnation point at $(x, y) = (0.5\,\mathrm{m}, 0.5\,\mathrm{m})$ is plotted in Figure 5.15.

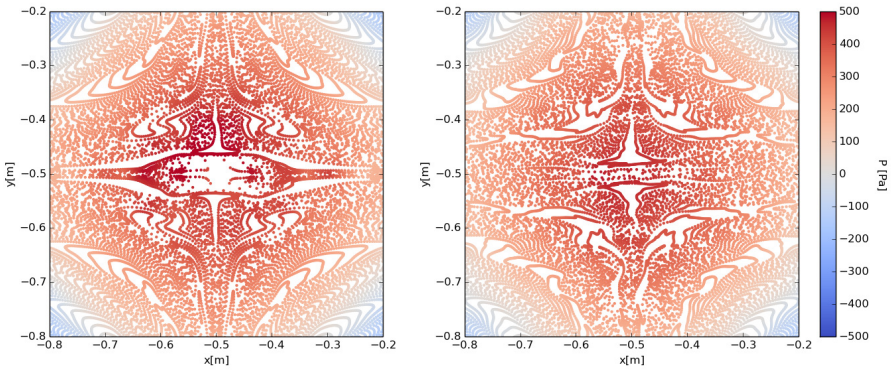


Figure 5.15: Particle distributions around the lower left stagnation point for the least square mapping (left panel) and the conservative mapping (right panel) at $t = 3\,\mathrm{s}$.

Important difference observed between both approaches is that the region near the stagnation points is filled with particles in the conservative mapping approach, whereas it is a hole in the least square mapping approach. This difference can be entirely attributed to the difference in the particle-to-grid mapping: in the conservative mapping the particle distribution is accounted for in the computation. This means that the element density is determined by the number of particles present in an element. As a result of this, overly filled elements will result in high density elements and hence a pressure gradient forcing the particles to low density elements. In this way, the particles are more or less forced to fill the entire domain, which is further confirmed when considering the nodal particle distribution. Ideally, it can be stated that the particle distribution as seen by the grid nodes remains constant during the entire computational cycle. In practice this would mean that if a particle leaves a gap somewhere, it is immediately filled with another particle. In mathematical terms, this condition was already formulated in Eq. (5.5). This equation provides also a means to perform a quantitative comparison of the particle distribution by defining the l2-error of the nodal particle distribution. This errornorm is given by:

$$\epsilon_{l2} = \sqrt{\sum_i \sum_p \left(N_i(\mathbf{x}_p(t)) - N_i(\mathbf{x}_p(t_0))\right)^2} \tag{5.6}$$

Plotting this error norm for the original implementation and for the MPM approach discussed in this section, results in Figure 5.16. A few things are noticeable in this figure. First of all, a quick increase in the particle error is observed at the beginning of the computation for both approaches. For the MPM mapping this quick increase in particle error is a result of the fact that initially the density field is far from being smooth (since the element density is determined by the number of particles in a cell). This is caused by the particle initialization procedure on a regular lattice, which results in a different number of particles per element and hence a non-uniform density field. The resulting density gradients will force particles from the overly filled and consequently high density areas to the under filled low density areas. Hence, the particle initialization itself will lead to a spurious motion at particle level. So the particle initialization itself is important for hybrid particle mesh methods. This problem is also recognized in literature. For the MPM approach for example, the problem can be circumvented by placing particles at the quadrature points and giving the particles the weight associated to the quadrature point weight [63]. As a result of this particle initialization, the first step in the scheme will have the same accuracy as a regular FEM method. The

issue of particle initialization is also recognized in full particle methods such as SPH. In e.g. [64] an SPH particle initialization algorithm was developed such that the initial spurious motions are oppressed as much as possible.
After the quick increase in particle distribution error, it is seen that the error remains approximately constant after $t/T \approx 1$ when using the conservative mapping. Compared to the original implementation, the particle distribution error is somewhat larger, although it is realized that the exact value of the particle distribution error is not that important but rather the boundedness of the error. In this regard, the conservative mapping performs better than the least square mapping. For the latter approach, strong wiggles are observed in the particle distribution error. Visual inspection of the particle distribution at these time instances (e.g. around $t/T \approx 1$) reveals that these wiggles coincide with the time instances that empty elements are present in the model domain. These empty elements are absent when using the conservative mapping.

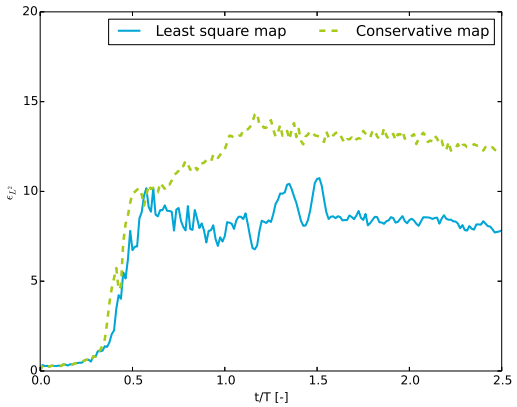


Figure 5.16: Error in particle distribution for the least square mapping and the conservative mapping.

Based on Figure 5.15 and Figure 5.16, the conclusion seems obvious: use the conservative mapping for the particle-to-grid mapping. Contrary to the least square mapping, this approach is both mass and (linear) momentum conserving. On top of that, a better control over the particle distribution can be achieved by using this method.

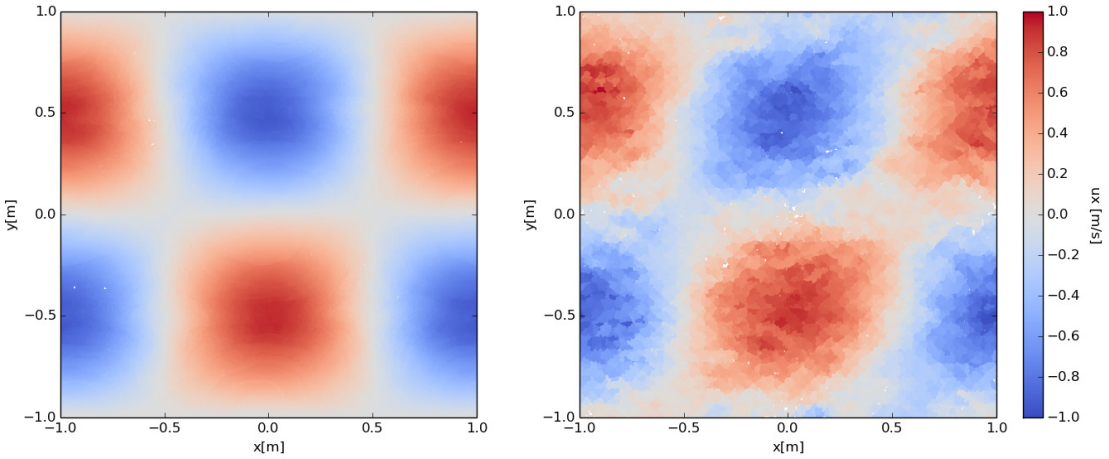


Figure 5.17: Horizontal velocity component for the least square mapping (left panel) and the conservative mapping (right panel) at $t = 7.5\,\mathrm{s}$

However, a drawback of the method becomes clear when running the model for an extended time range. In e.g. Figure 5.2 and Figure 5.3 it was seen that the simulated pressure and velocity profiles are

well maintained during the simulation employing the least square mapping. This is however not the case when using the conservative mapping. As an example, consider Figure 5.17 in which the horizontal velocity component at particle level at $t = 7.5\,\mathrm{s}$ is plotted for both the least square mapping and the conservative mapping. Contrary to the least square mapping, the resulting velocity field for the conservative mapping becomes extremely distorted. This is expected to be the result of the drawback of the MPM approach which was already mentioned in Section 3.1: the MPM approach does not make use of an optimal quadrature, which consequently results in a noisy integration and hence a flow field prone to distortions. Although it can be expected that this problem can be mitigated by using higher order elements, using such elements does not solve the underlying fundamental issue.
Based on the comparison of the least square mapping and the conservative mapping, it can be concluded that although the latter mentioned approach leads to better particle distributions and is both mass and momentum conserving, it does lead to a distorted flow field which can be attributed to the way in which numerical integrals are approximated.

5.2 Lid driven cavity flow

5.2.1. Problem description

As a second example for one phase flows, the lid driven cavity problem is tested. This test is extensively used as a benchmark for assessing new solution procedures. The problem set-up is straightforward: a fluid in a square cavity with closed, no-slip boundaries is set in motion by a tangential velocity at the top of the square box. Due to this tangential velocity boundary combined with the viscous shear forces, an eddy is generated in the square cavity.
Specific to the hybrid particle mesh method, the lid driven cavity problem is a useful test case to assess different features of the developed method. First of all, it provides insight in the implementation of the viscous term as presented in Section 5.1.5. Secondly, it provides insight in the no-slip condition (enforced in a weak sense). Whereas in the Taylor-Green test the boundary term in Eq. (4.27) could be omitted because only periodic boundary conditions were present, the implementation of the boundary term is all-important for the lid driven cavity problem, since this term is forcing the flow in the cavity. Secondly, it can be anticipated that the lid driven cavity problem is an extreme test case for assessing the quality of the particle distribution due to the stagnating character of the flow. This will especially be the case for advection-dominated problems. Therefore, two cases are considered in this section: a test case using a low Reynolds number ($\mathrm{Re} = 1$) in which the advective term plays a minor role and a test case for a slightly higher Reynolds number ($\mathrm{Re} = 100$) in which the advective term starts to dominate the viscous term. For the lid driven cavity problem, the Reynolds number Re is defined as:

$$\mathrm{Re} = \frac{UD}{\nu} \tag{5.7}$$

in which U is the tangential velocity at the top, D is a characteristic length for the square cavity and ν is the kinematic viscosity.
Results for both test cases are compared to values found in literature: for the $\mathrm{Re} = 1$ test case, MPM solutions as presented by Hamad [5] as well as SPH and finite volume (FVM) solutions as presented by de Wit [65] are used as a reference. For the $\mathrm{Re} = 100$ test case, SPH solutions as presented by Xu *et al.* [14] are used as reference.

5.2.2. Model configuration

Basic model settings are listed in Table 5.2. For the two different test cases only the kinematic viscosity ν and the timestesp Δt remains to be specified.

Table 5.2: Model properties lid-driven cavity problem.

Model parameters	Ω	$[0,1]^2$m	Computational domain
	D	1 m	Characteristic length
	nx×ny	40×40	Cells in x- and y-direction (diagonal type)
	Δx_p	0.0025m	Particle spacing
	α	0.02	PIC fraction
	U	1 ms^{-1}	Tangential velocity top
	Δt	Case dependent	Time step
	ρ	1000 kgm^{-3}	Density
	ν	Case dependent	Kinematic viscosity
Computational aspects		160 000	Number of particles
		50	Average nr. particles per element

5.2.3. Low Reynolds number test

The first test case to be run is for a very low Reynolds number of 1. This is achieved by setting $\nu = 1\,\text{m}^2\text{s}^{-1}$. A timestep $\Delta t = 1E-5\,\text{s}$ is used. This value was obtained by trial and error: for larger timesteps the computation became instable.

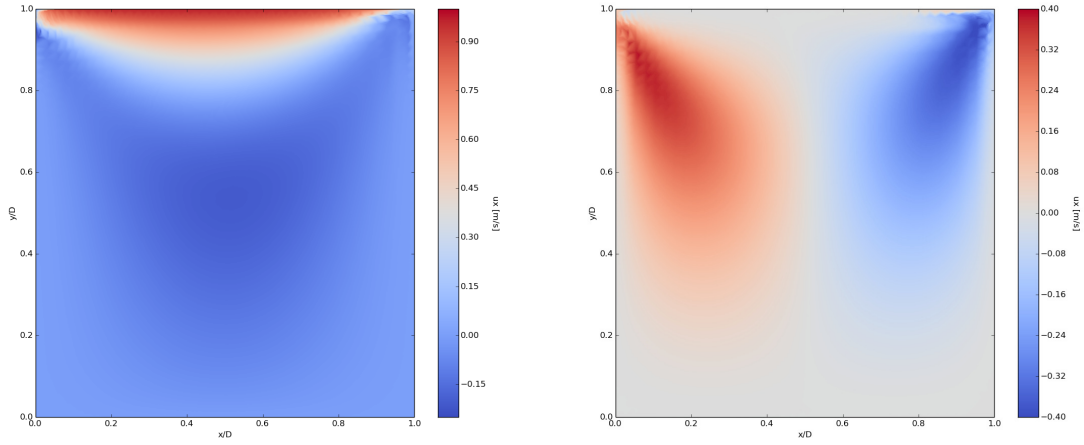


Figure 5.18: Horizontal (left panel) and vertical (right panel) particle velocities for the Re = 1 lid driven cavity problem.

The resulting velocity fields in $x-$ and $y-$direction for the Re = 1 test case at $t = 0.1\,\text{s}$ are depicted in Figure 5.18. At this time instance, the problem converged to a 'steady state' [2]. The velocity is evaluated at particle level, i.e. the depicted fields are particle velocity fields. Qualitatively, the result is as expected: a high velocity in positive x-direction near the top boundary and a return flow in negative x-direction is observed in the lower part of the cavity. To close the vortex in the cavity, the fluid sinks at the right boundary and rises at the left boundary. Furthermore, no irregularities are observed in the particle distribution for this test case: the whole domain remains filled with particles. It must however be kept in mind that a low Reynolds number test case is relatively easy, since the particle advection is almost negligible and hence the particle distribution is approximately similar to the initial particle distribution.

[2] The term 'steady state' should be understood correctly in the context of (hybrid Eulerian-) Lagrangian methods. Specific to hybrid particle-mesh methods it means that a steady state solution is obtained at the *background grid*, this however does not yield at *particle level* as the particles move in and interact with the non-uniform velocity field at the background grid.

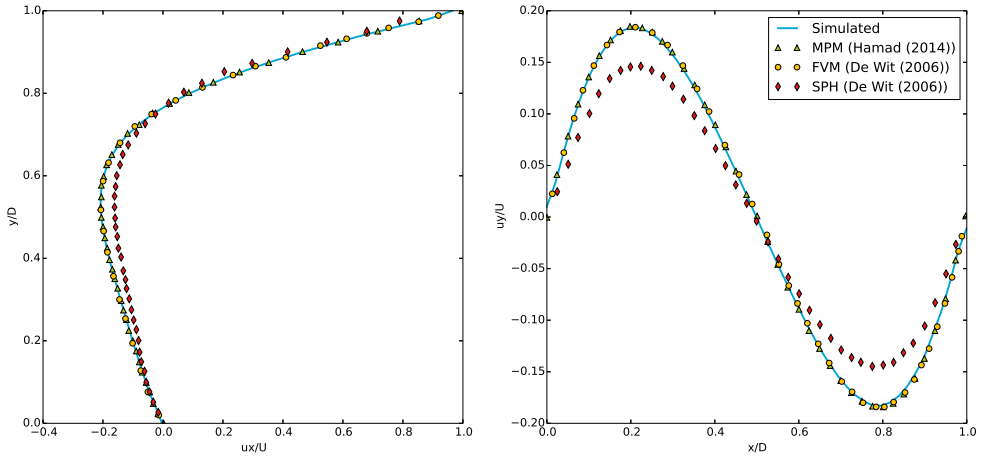


Figure 5.19: Normalized horizontal (left panel) and vertical (right panel) velocities along the cavity centerlines for Re = 1.

Model results are quantitatively assessed by comparing the results to MPM literature values as presented in [5] and SPH and FVM results as presented in [65]. An additional note on the MPM results should be made: for the MPM test case the particles are assumed to be fixed in place. Combined with an initial particle placement at quadrature points, the MPM approach thus effectively reduces to regular FEM. The assumption of fixed particles is approximately valid for low Reynolds numbers but is obviously not so for higher Reynolds numbers in which the problem is advection dominated. In fact, keeping the particle locations fixed neglects the key feature of advecting field properties with particles. Thus, fundamental differences can be expected between low and high Reynolds number tests for the MPM approach.
Model results and literature values for the low Reynolds number test case are compared in Figure 5.19. In this figure the horizontal and vertical velocities along respectively the vertical and horizontal centerline are plotted. Simulated results and results obtained with MPM and FVM are in perfect agreement with each other, showing that the method developed in this thesis is able to accurately reconstruct the results obtained with well-established methods. Furthermore, the developed model does predict the velocity profiles more accurately than the SPH method presented in [65].
Small differences are however observed at the boundaries. Whereas the MPM and FVM results exactly satisfy the no-slip conditions, these conditions are only approximately satisfied by the developed model. This is the result of enforcing the boundary conditions only in a weak sense (see Eq. (4.27)) instead of strongly imposing the boundary conditions as done in MPM.

5.2.4. Increased Reynolds number test

Unfortunately, Hamad [5] and de Wit [65] only presents results for very low Reynolds numbers, whereas it has been argued in the previous paragraph that high Reynolds number test cases are much more demanding for (hybrid) particle methods. Therefore, the lid driven cavity problem is simulated again but now for Re = 100. The kinematic viscosity is correspondingly set to $\nu = 100\,\mathrm{m^2 s^{-1}}$, the timestep was set to $\Delta t = 1E-3\,\mathrm{s}$. All other model parameters are defined in Table 5.2. The solution converged to a steady state solution at approximately $t = 5\,\mathrm{s}$.
The horizontal and vertical velocities at $t = 5\,\mathrm{s}$ are compared with literature values as presented in [14] where an SPH implementation is used to simulate the lid-driven cavity problem, see Figure 5.20. Compared to the Re = 1 testcase, the differences between literature values and simulated values are slightly larger although results are still in very good agreement with each other, where it is again noted that the no-slip boundary conditions are not exactly satisfied in the simulations.
The results for the Re = 100 test case seems encouraging, nevertheless fundamental problems are observed when plotting the particle distributions at $t = 5\,\mathrm{s}$, see Figure 5.21. In this figure it can be clearly seen that the particle distribution becomes highly distorted after a certain model runtime. Gaps in the particle distrbution are especially observed in the top right corner where the flow stagnates and near the right

boundary where the eddy separates from the boundary. Obviously, the present model implementation cannot correctly deal with these issues. The underlying reason for this has been discussed in Section 4.2 and Section 5.1.6: the present model implementation does not offer any control on the particle distribution, a drawback of which the ramifications become clear in stagnating flows such as the lid-driven cavity problem. For the time being, no additional measures are taken in order to mitigate this problem, nevertheless it is clear that the lack of control over particle positions is a clear disadvantage of the present implementation and needs further attention in future research.

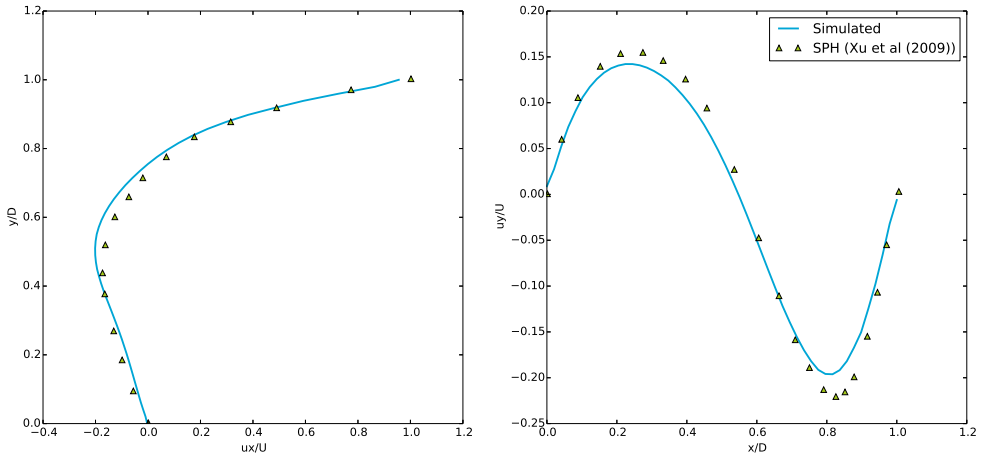


Figure 5.20: Normalized horizontal (left panel) and vertical (right panel) velocities along the cavity centerlines for Re = 100.

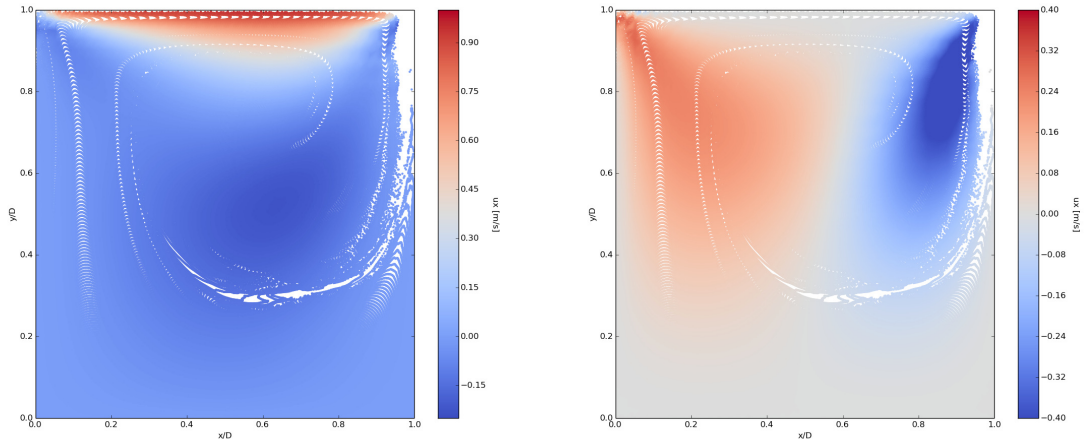


Figure 5.21: Normalized horizontal (left panel) and vertical (right panel) velocities along the cavity centerlines for Re = 100.

5.3 Summary and conclusions

Two single-phase flow benchmark tests were performed for the developed model implementation to gain insight in the performance of the method. By comparing model results for the Taylor-Green vortex problem to analytical solutions, it became clear that a certain amount of numerical diffusion is introduced by the method, resulting in a slight damping of velocity and pressures. Relating the numerical diffusion to the physical diffusion, revealed that the numerical viscosity for the given model set-up is approximately an order of 10^2 larger than the viscosity of water. It was shown that the amount of numerical diffusion turned out to be strongly dependent on the PIC fraction, where a higher PIC fraction leads to more numerical diffusion. Regarding the Taylor-Green vortex it can be concluded that the model is able to maintain the velocity and

pressure profiles very well, even on irregular grids. For a different particle-to-grid mapping procedure, being equal to a simple MPM implementation, noisy results were observed, which leads to strongly distorted flow fields.
The implementation of the viscous term, including no-slip boundary conditions, was assessed by running the lid driven cavity benchmark. For a low Reynolds number test a perfect agreement was obtained between results obtained with the developed model and MPM and FVM results, whereas SPH results were shown to be somewhat less accurate. However, ramifications of the model implementation became clear when running the lid driven cavity benchmark for an increased Reynolds number. In this advection dominated flow, the particle distribution became extremely disordered, resulting in gaps in the model domain corners and along the boundaries of the domain. This unwanted behavior (in fact a violation of the incompressibility constraint at particle level) was attributed to the particle-grid interaction process and it was concluded that this issue certainly needs attention in future research.

6

Numerical Testcases for two-phase flows

In the previous chapter some academic single-phase flow test cases were considered. This chapter will make the step to two-phase fluid simulations. The first test case of the Rayleigh-Taylor instability will reveal the performance of the scheme for a density ratio close to unity. After this, the method is assessed for different problems involving large density ratios (i.e. the water/air density ratio of 1000). These benchmarks become more of practical importance since a two-phase implementation allows the analysis of wave propagation processes involving an air-water interface. Purpose of running such benchmark tests is thus to assess whether the method is well-suited to set-up a numerical wave flume. To this end, a wave generating boundary condition is implemented and results are compared to analytical values for a solitary wave. As a final example, the simulation of wave breaking with the hybrid particle-mesh method is qualitatively assessed.

6.1 Rayleigh-Taylor instability

As a first test case for two-phase fluid flows, the numerical simulation of a Rayleigh-Taylor instability is analyzed. Since this problem typically involves small density ratios, the problem can be regarded as an intermediate step between the single-phase simulations and the high-density ratio two-phase flow simulations. Main purpose of doing this test case is to see if the interface separating the two phases is well-maintained and if the interface propagation is well-captured by the model.
The Rayleigh-Taylor instability typically occurs when a heavy fluid is initially placed on top of a lighter fluid. The growth of this interface instability was first studied by Rayleigh, who derived for the linearized case an exponential growth of the interface disturbance for inviscid cases as:

$$\zeta(x,t) = \zeta_0(x) e^{\sqrt{gkA}t} \tag{6.1}$$

In this equation, $\zeta(x,t)$ is the interface position and $\zeta_0(x)$ the initial interface position. Furthermore, k denotes the spatial wave number of the initial perturbation and A is the so-called Atwood number, defined as:

$$A = \frac{\rho_2 - \rho_1}{\rho_2 + \rho_1} \tag{6.2}$$

where ρ_1 and ρ_2 are the density of the bottom and the top layer respectively. It is clear that the instability grows for $\rho_2 > \rho_1$, i.e. when the heavy fluid is on top of the lighter fluid.
Simulation of the Rayleigh-Taylor stability was done in a closed domain $\Omega = [0, 1.5] \times [0, 2]$m discretized by means of a crossed, regular triangular grid and using an initial sinusoidal disturbance:

$$\zeta_0(x) = H \sin(kx) \tag{6.3}$$

An overview of the relevant parameters is listed in Table 6.1.

Table 6.1: Model settings Rayleigh-Taylor instability

Model parameters	Ω	$[0, 1.5] \times [0, 2.0]$m	Computational domain
	nx×ny	60×100	Cells in x- and y-direction (crossed)
	Δx_p	0.002m	Particle spacing
	α	0.02	PIC fraction
	H	0.05 m	Amplitude initial disturbance
	Δt	$5E-4$ s	Time step
	A	0.5	Atwood number
	ν	0 m^2s^{-1}	Kinematic viscosity
	k	$\frac{2*\pi}{0.5}$	Wave length
Computational aspects		750 000	Number of particles
		31	Average nr. particles per element
		1.2 s	Simulated physical time
		52 min	Model run time

The instability evolution is depicted in Figure 6.1 for the disturbance in the middle of the domain. As expected, the initially small disturbance grows in time. The downward moving spike of heavy fluid rolls up in two counter-rotating vortices. It can be expected that these large vortices break again into smaller vortices, but this process is barely observable. Reason for this might be the dissipative nature of the scheme, oppressing small scale features.

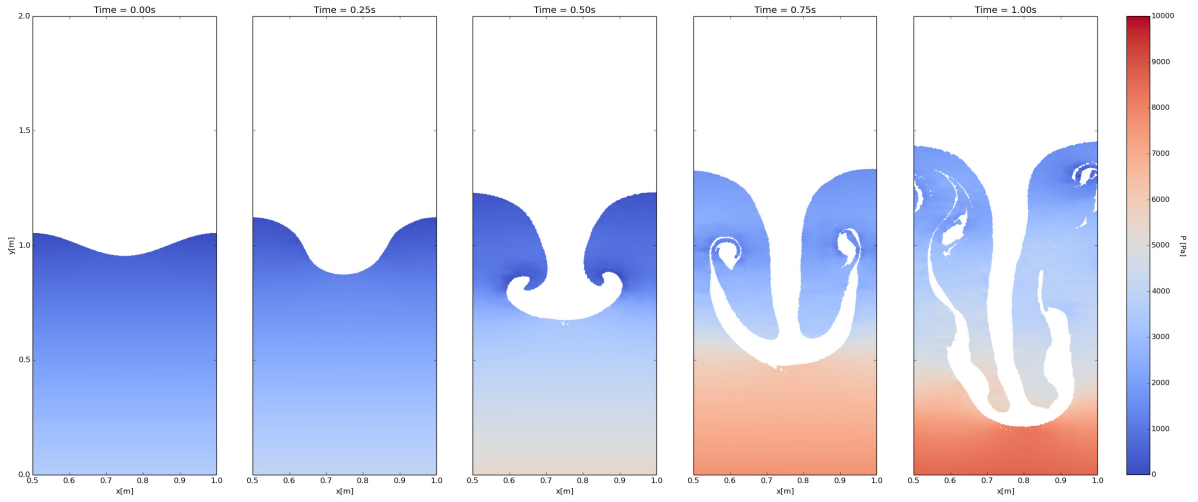


Figure 6.1: Evolution of a Rayleigh-Taylor instability for an Atwood number of 0.5.

Bubble and spike velocities are compared to results found in literature ([66], [67]). In the first mentioned paper an inviscid fluid is considered, whereas viscosity is taken into account in the second paper, using $\mathrm{Re} = 256$ and $\mathrm{Re} = 2048$, with Re is defined as:

$$\mathrm{Re} = \frac{\sqrt{LgL}}{\nu} \tag{6.4}$$

and $L = \frac{2\pi}{k}$ the wave length.
The influence of the viscosity on the macroscopic behavior of the instability can be assumed to be negligible for the high Re case, therefore, only an inviscid simulation is run.
The simulated position and velocity of the bubble and the spike are depicted in Figure 6.2 and Figure 6.3 respectively. According to [67] a time scale of $T = \sqrt{Lg^{-1}}$, a length scale of L and a velocity scale $\sqrt{AgL}$

are used to non-dimensionalize the axes in these figures. A good agreement is observed for the simulated bubble and spike positions and the literature values. The simulated velocities, especially the bubble velocity deviates slightly from the literature values although the pattern of the profile is the same. Two remarkable things are observed in Figure 6.3: first, the dimensionless spike velocity reaches a constant cruising speed of approximately 0.3 which matches closely with the reported values of 0.27 [67] and 0.265 [66]. Second, after an initial acceleration the simulated spike velocity decelerates around $\frac{t}{T} \approx 2$, from approximately $\frac{t}{T} \approx 3$ the spike accelerates again. A feature which is also observed in literature. As this pattern is reported to be less pronounced for lower Reynolds numbers, He *et al.* [67] attributed the phenomenon to the secondary vortices, which are only present for high Reynolds numbers.

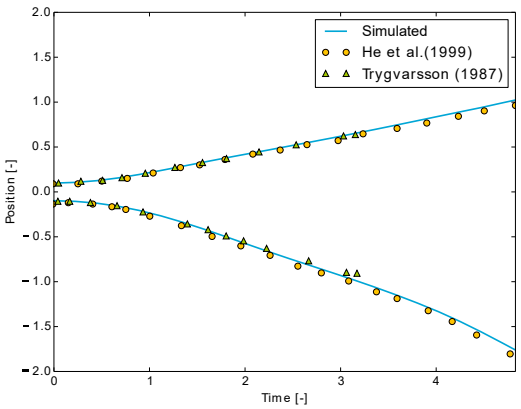


Figure 6.2: Simulated bubble and spike position of the developing Rayleigh-Taylor instability compared to literature values.

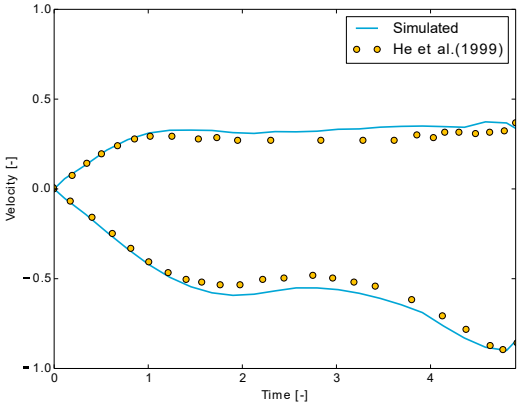


Figure 6.3: Simulated bubble and spike velocity of the developing Rayleigh-Taylor instability compared to literature values.

The energy present in the system is computed for different PIC fractions. As the (inviscid) Rayleigh-Taylor instability test case forms a closed system, without additional sources or sinks, the total energy should be conserved. Only a conversion from potential into kinetic energy should take place. This conversion process can be considered at the grid level (by defining the energy to be a grid quantity) or the potential and kinetic energy can be considered to be a particle feature. Choosing the latter option, potential and kinetic energy are respectively computed as:

$$E_\mathrm{p} = \sum_p m_p g z \tag{6.5}$$

$$E_\mathrm{k} = \sum_p \frac{1}{2} m_p \|\mathbf{u}_p\|^2 \tag{6.6}$$

with E_k the kinetic energy and E_p the potential energy.
In order to visualize the energy conversion process, the potential and kinetic energy changes are cast in dimensionless form as:

$$\Delta E_\mathrm{p} = \frac{E_\mathrm{p}(t) - E_\mathrm{p}(t^0)}{E(t^0)} = \frac{E_\mathrm{p}(t) - E_\mathrm{p}(t^0)}{E_\mathrm{p}(t^0)} \tag{6.7}$$

$$\Delta E_\mathrm{k} = \frac{E_\mathrm{k} - E_\mathrm{k}(t^0)}{E(t^0)} = \frac{E_\mathrm{k}(t)}{E_\mathrm{p}(t^0)} \tag{6.8}$$

Since no physical diffusion is present in the system, the sum of Eq. (6.7) and Eq. (6.8) should add up to 0 in order to have an energy conserving scheme. This is obviously not the case as a slight decay in the total energy is observed in time. This decay can be partially attributed to the PIC fraction (controlled by α), as a stronger decay is visible for higher PIC fractions. Unfortunately, the model became unstable for a PIC fraction of 0.0, so that the energy decay for a pure FLIP computation could not be investigated for this two-phase test case.

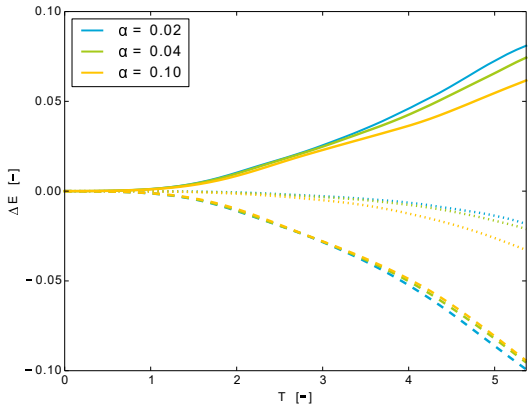


Figure 6.4: Kinetic (solid), potential (dashed) and total energy (dotted) for different PIC fractions over time.

The Rayleigh-Taylor instability reveals that the developed scheme is able to simulate two-phase flows in which the densities of the two-phases is of the same order of magnitude. The sharp interface between the two phases is well-maintained and it was shown that bubble and spike displacements and velocities for the Rayleigh-Taylor instability are in good agreement with literature values. Not surprisingly, the value of the PIC fraction determines to a large extent the amount of numerical dissipation. Finally, it is noted that the PIC fraction was set to a small, positive number in order to obtain stable model runs.

6.2 Standing Wave

As a second two-phase benchmark, the standing wave in a closed basin is considered. Compared to the previous described Rayleigh-Taylor test case, the density difference between the two phases is taken much bigger. In fact, the air-water density ratio of approximately 1000 is used. The standing wave test case will thus reveal whether the scheme is able to deal with those large density differences.
Apart from this, the standing wave test case will provide useful information on the numerical diffusion and numerical phase errors. In the absence of viscosity and using free-slip boundary conditions, the energy in the system should stay constant. On top of that, if the scheme shows a numerical phase error, simulated and analytical wave periods will differ.
Analytical expressions for the free surface elevation and the wave period can be derived from the linear wave theory. This theory holds under the small-amplitude approximation, i.e. the wave amplitude must be small with respect to both the wave length and the water depth. When these requirements are satisfied, the free surface elevation is given by:

$$\zeta(x,t) = H\cos(kx)\cos(\omega t) \tag{6.9}$$

with $\zeta(x,t)$ the location of the air-water interface, H the wave amplitude, k the wave number and the radian frequency ω computed from the dispersion relationship:

$$\omega^2 = gk\tanh(kd) \tag{6.10}$$

in which d the still water depth.
For the test case, the parameters as listed in Table 6.2 are used. Initial conditions are provided by a cosine shaped free surface profile, with a wave length equal to the basin length (i.e. $\lambda = L$) and an amplitude/depth ratio of 0.1, which can be considered to be an upper limit for the small amplitude approximation to be valid. Reason for choosing a relatively large amplitude/depth ratio is that the wave amplitude must be several times larger than the characteristic length scale for the elements at the background grid. As a result of the relatively large amplitude/depth ratio, the simulated wave profile may contain some non-linear effects. Nevertheless these non-linear effects are ignored so that simulated results can be compared to analytical results obtained from the linear wave theory.
A special note should be made regarding the time step, which is set to $\Delta t = 1E-3\,\mathrm{s}$. Compared to a

comparable problem using a compressible MPM formulation [68], the time step in the present formulation is approximately an order 10^2 larger than in the compressible formulation.

Table 6.2: Model settings standing wave.

Model parameters	Ω	$[0, 20] \times [0, 10]$m	Computational domain
	nx×ny	80×40	Cells in x- and y-direction (diagonal)
	Δx_p	0.02 m	Particle spacing
	α	0.02	PIC fraction
	d	5 m	Undisturbed water depth
	H	0.5 m	Wave amplitude
	Δt	$1E-3$ s	Time step
	ν	$0\,\mathrm{m^2 s^{-1}}$	Kinematic viscosity
	k	$\frac{2*\pi}{20}$	Wave length
Computational aspects		500 000	Number of particles
		80	Average nr. particles per element
		15 s	Simulated physical time
		176 min	Model run time

As a first test, the basic scheme as presented in Section 4.7 is used. The velocity profile after one timestep is presented in Figure 6.5. Around the free surface, spurious velocities are observed. This is a strictly numerical artifact, caused by the way the density gradient is computed (see e.g. [69]). For a better understanding of this problem, consider the balance between the pressure gradient term and the gravitational body force in equilibrium conditions, i.e.:

$$\frac{1}{\rho}\nabla p = \mathbf{g} \tag{6.11}$$

In a two-phase fluid problem, the density term however changes discontinuously at the interface. As long as the interface is aligned with the background mesh, this discontinuity can be handled if ρ changes discontinuously between elements. However, when the air-water interface cuts through an element, equilibrium between the pressure gradient term and the body force cannot simply be achieved element-wise. This is essentially the result of the ambiguous definition of the density in such cells, a feature which expresses itself in the spurious interface velocities [69].
As a way of *mitigating* this problem, it is recognized that although the pressure gradient itself and the density itself are discontinuous at the interface, the term $\frac{1}{\rho}\nabla p$ is continuous in gravitational equilibrium equaling the gravity term $\mathbf{g}$, provided that surface tension effects are neglected. Inspired by this notion, the pressure gradient term $\frac{1}{\rho}\nabla p$ is projected to a piecewise linear field. In other words, find $\mathbf{r} \in \mathcal{R}$ such that:

$$\int_\Omega \mathbf{r} \cdot \mathbf{s} \mathrm{d}\Omega = \int_\Omega \frac{1}{\rho}\nabla p \cdot \mathbf{s} \mathrm{d}\Omega \qquad \forall \mathbf{s} \in \mathcal{S} \tag{6.12}$$

in which the test and trial function spaces are respectively given by:

$$\mathcal{R} = \{\boldsymbol{r} \in \mathbf{H}^1 : \boldsymbol{r} \in \mathbf{P}^1(\Omega_e) \ \forall e\} \tag{6.13}$$

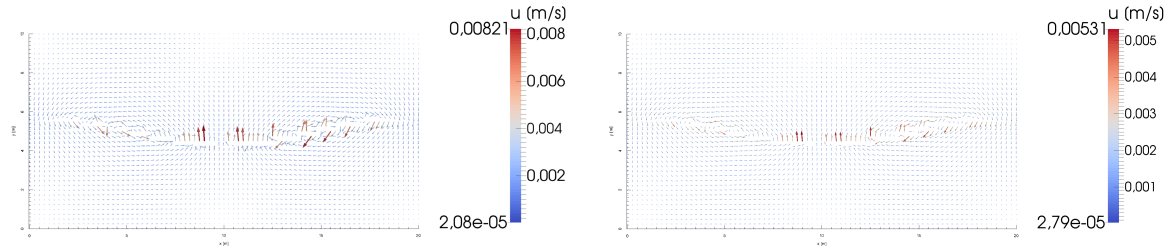
$$\mathcal{S} = \{\boldsymbol{s} \in \mathbf{H}^1 : \boldsymbol{s} \in \mathbf{P}^1(\Omega_e) \ \forall e\} \tag{6.14}$$

In addition to this, the end-of-time step velocity (Eq. (4.23)) is modified to: find $\mathbf{u}_h^{n+1} \in \boldsymbol{\mathcal{V}}_h$ such that

$$\int_\Omega \mathbf{u}_h^{n+1} \cdot \mathbf{v}_h \mathrm{d}\Omega = \int_\Omega \mathbf{u}_*^{n+1} \cdot \mathbf{v}_h \mathrm{d}\Omega - \Delta t \int_\Omega \mathbf{r}^{n+1} \cdot \mathbf{v}_h \mathrm{d}\Omega \qquad \forall \mathbf{v}_h \in \boldsymbol{\mathcal{V}}_h \tag{6.15}$$

The difference between the original scheme and the scheme with the above derived modifications, becomes clear when considering Figure 6.5. In the left panel quite severe velocity oscillations are observed, whereas a more regular velocity field is obtained (right panel) when projecting the pressure gradient term to piecewise

linears as presented above.



(a) Original scheme. (b) Modified scheme.

Figure 6.5: Velocity vectors in standing wave after one timestep. Note the different color scaling.

It is thus concluded that the spurious velocities can be attributed to the density jump at the interface. These spurious velocities are oppressed by using a different way of computing the pressure gradient at the background grid. It is however emphasized that this approach is certainly not a fundamental solution to the problem. Therefore, a strategy should be developed in future research in order to solve this problem. In this regard the extended finite element method (XFEM) may be a possible candidate for solving the interface issues. In this technique the basis functions are enriched in such a way that they can deal with sharp discontinuities [70]. Another approach would be to consider the diffuse-interface methods, in which the interface is assumed to have a finite width, thus avoiding sharp discontinuities. For the diffuse-interface approach see e.g. [71].
The resulting pressure fields for the water-phase and the air-water interface profile are depicted in Figure 6.6 at two different time instances, revealing that the resulting pressure fields are perfectly smooth and no pathological locking effects are observed. The air-water interphase is relatively well maintained over time, although small free surface disturbances are observed near the nodes of the cosine wave (visible in the right panel of Figure 6.6). These irregularities are still a result of small spurious velocities occuring near the interface.

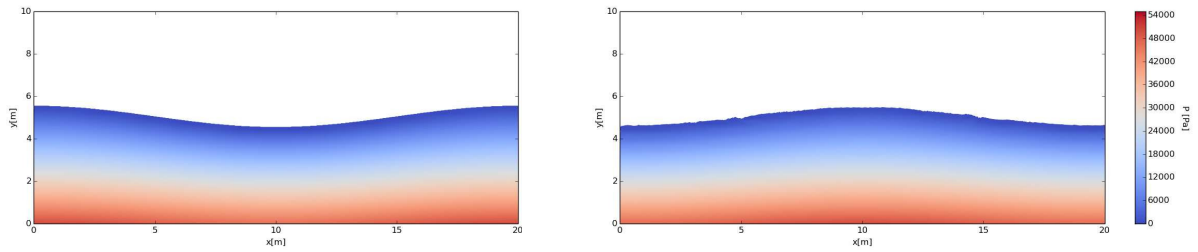


Figure 6.6: Pressures in standing wave with $H/d = 0.1$ at $t = 0\,\mathrm{s}$ and $t = 5.5\,\mathrm{s}$.

A quantitative comparison of the simulated amplitudes to the analytical amplitudes at $x = 0.5L = 10\,\mathrm{m}$ results in Figure 6.7. An amplitude decay is observed, confirming the findings in the Taylor-Green and the Rayleigh-Taylor test case that some numerical diffusivity is present in the scheme. This artifact was shown to be enhanced by using strictly positive PIC fractions (a value $\alpha = 0.02$ was used for the standing wave test).
The simulated amplitudes are qualitatively compared to an SPH simulation of a standing wave [65]. In this reference, an amplitude/depth ratio of 0.075 was used, combined with a particle resolution of approximately 160 particles/wavelength and 6 particles over the wave amplitude. The particle resolution in SPH is approximately similar to the resolution of the background grid in the hybrid particle-mesh implementation, although it must be mentioned that the particle resolution used in the simulation performed in this thesis

is much higher (Table 6.2).
A qualitative comparison between the two methods reveals the amplitudes to be better maintained in the hybrid particle-mesh implementation compared to the SPH method in which the amplitude decay was approximately 50% in three wave periods, compared to an amplitude decay of approximately 20% in the hybrid particle-mesh implementation over the same number of wave periods.
Apart from the amplitude decay, the simulated results show a phase lag with respect to the analytical result. This is also expected to be the result of the diffusivity in the system, since diffusion will tend to decrease the wave celerity c. With $\omega = ck$, and k constant, it follows that ω will decrease for decreasing c and hence the wave period $T = \frac{2\pi}{\omega}$ will increase. Numerical diffusion will thus lead to simulated solutions lagging behind the analytical solution.
The diffusive character of the scheme is also observed when plotting the potential and the kinetic energy, respectively computed at particle level with Eq. (6.5) and Eq. (6.6). As expected, potential energy maxima coincides with kinetic energy minima, showing that in a standing wave potential energy is repeatedly converted to kinetic energy and vice versa. A clear decay in total energy is observed: approximately 40% in four wave periods. Compared to SPH, such a value is not even too bad: an energy decay of 69% in four wave periods was for example reported by de Wit [65].
As a final comment regarding the energy plot, it is observed that the minimum potential energy decreases over time. Such a behavior was also observed in the SPH method of de Wit [65], although no clear explanation was given by this author. This artifact is however expected to be the result of a residual settling of the fluid particles. A feature which is possibly the consequence of lacking a proper balance between the pressure gradient term and the body force at particle level (Eq. (6.11)).

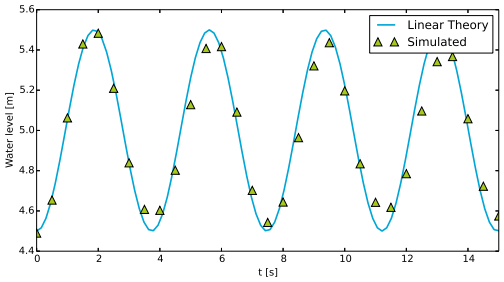


Figure 6.7: Theoretical and simulated surface elevation at $x = 0.5L$.

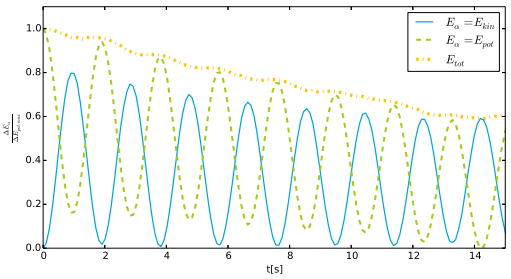


Figure 6.8: Total, kinetic and potential energy in standing wave.

Finally, the computed pressures are compared to the analytical pressures. To this end, the wave-induced pressure is considered. For linear waves, the total pressure is given by [72]:

$$p = -\rho g z + \rho g H \frac{\cosh\left[k(d+z)\right]}{\cosh(kd)} \sin(\omega t - kx) \tag{6.16}$$

Where it is noted that $z = 0$ at the still water depth with z assumed to be positive upwards.
The first term at the right hand side represents the hydrostatic balance, whereas the second term represents the wave-induced pressure, i.e.:

$$p_{\text{wave}} = \rho g H \frac{\cosh\left[k(d+z)\right]}{\cosh(kd)} \sin(\omega t - kx) \tag{6.17}$$

Simulated wave induced pressures are computed by subtracting the term $\rho g z$ from the total pressure (and assuming $z = 0$ at the still water depth d). The resulting values are plotted against the analytical values following from the linear wave theory (Eq. (6.17)) at $x = 0.5L = 10\,\text{m}$ after a half wave period ($t \approx 1.7\,\text{s}$) and a full wave period ($t \approx 3.8\,\text{s}$). Results are depicted in Figure 6.9, showing a very good correspondence both at a time instance of maximum elevation (left panel) and minimum elevation (right panel). The slight underestimation of the wave-induced pressure for the maximum elevation of approximately 500 Pa should be put in perspective by noting that this difference is only around 1% of the maximum pressure (see Figure 6.6)

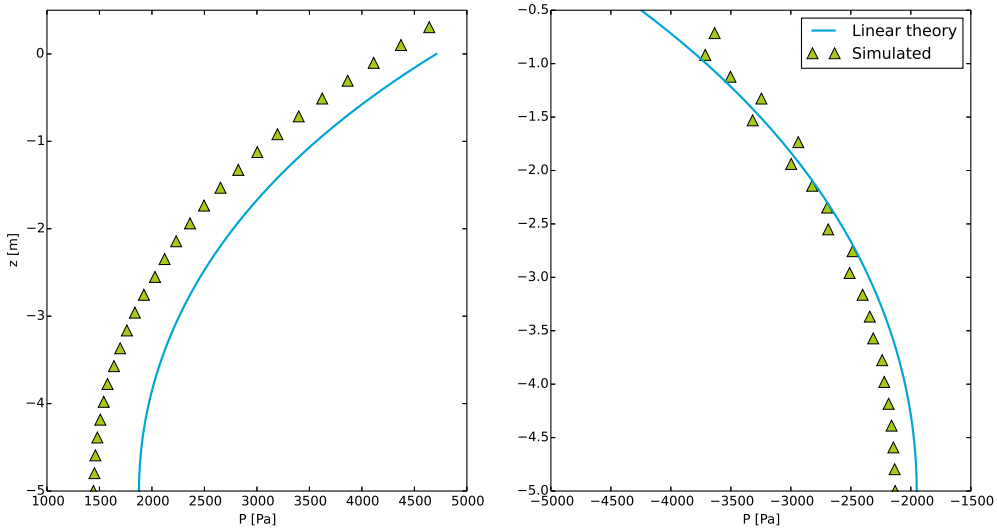


Figure 6.9: Wave-induced pressure in standing wave at $x = 0.5L$ at $t = 1.7\,\mathrm{s}$ and $t = 3.8\,\mathrm{s}$.

6.3 Solitary Wave

As a test case to assess the performance of the scheme for large amplitude problems, the propagation of a solitary wave over a flat bed is considered. In solitary waves, the non-linear effects exactly balance the wave dispersion [73]. In order to understand this, one can think of the non-linear process to be a continuous injection of short-wavelengths at the leading edge of the main pulse, which would normally lead to a steepening of the front and a flattening of the trailing edge. However, given the dispersion relation, the celerity of the short-wavelength components is smaller than the celerity of the main pulse, so that the short-wavelength components are carried off to the back of the pulse. In a solitary wave, these two processes balance each other, resulting in a stable pulse preserving shape and form. For solitary waves in free surface flows, the surface elevation ζ can be approximated by:

$$\zeta = H\mathrm{sech}^2\left[\sqrt{\frac{3H}{4d^3}}(x - ct)\right] \tag{6.18}$$

with H the wave height, d the undisturbed water depth and $c = \sqrt{g(d+H)}$ the wave celerity. Furthermore, horizontal and vertical velocity components are respectively approximated as:

$$u = c\frac{\zeta}{\zeta + d} \tag{6.19}$$

$$v = -(z+d)\frac{\partial u}{\partial x} \tag{6.20}$$

where z is the vertical coordinate, being zero at the undisturbed water depth d. The above given solution is accurate to order $(H/d)^2$ [74].

For this test case, a rectangular flume of length 50 m and total height 2 m is used. The undisturbed water depth is set to $d = 1\,\mathrm{m}$, initial conditions are provided by Eqs. (6.18)-(6.20), the wave crest is initially positioned at $x = 10\,\mathrm{m}$. When imposing the initial conditions, a difficulty arises as the initial conditions are only provided for the water phase, whereas in the two-phase approach initial conditions are also needed for the air phase. These initial conditions are obtained iteratively by solving the set of equations using the initial conditions for the water phase and updating the velocity in the air region. In the next iteration steps, the initial conditions for the water phase are combined with the updated velocities in the air region, until the resulting (velocity) solution between consecutive iteration steps is sufficiently converged.

Model parameters are listed in Table 6.3. The model time step $2.5E-3\,\mathrm{s}$ corresponds to a CFL number of approximately 0.05. Computations are run for 7 s of physical time. One important remark to be made,

is that the computational grid is aligned with the location of the undisturbed water depth. According to the discussion in the previous section, the jump in density is easily dealt with in such a configuration.

Table 6.3: Model properties solitary wave propagation.

Model parameters	Ω	$[0, 50] \times [0, 2.0]$m	Computational domain
	nx×ny	200×20	Cells in x- and y-direction (diagonal)
	Δx_p	0.01 m	Particle spacing
	α	0.02	PIC fraction
	Δt	$2.5E-3$ s	Time step
	d	1.0 m	Still water depth
	ν	0 m^2s^{-1}	Kinematic viscosity
Computational aspects		1000000	Number of particles
		125	Average nr. particles per element
		7.5 s	Simulated physical time
		87 min	Model run time

The initial pressure field and the pressure field at $t = 7$ s are shown in Figure 6.10 for a relative wave height of $H/d = 0.4$. Although not further analyzed quantitatively, it is clear that the resulting pressure field remains perfectly smooth, again revealing that pathological locking effects are absent using the developed approach. Another feature observed in the figure for $t = 7$ s is that the free surface at the trailing edge is not entirely restored to equilibrium, instead, small oscillations are observed around the undisturbed water depth. These oscillations are the result of a fundamental problem in our present approach which was already mentioned in preceding sections: static equilibrium can neither be achieved at element level for the general case that an interface cuts through an element, nor can it be obtained at particle level. So even when considering the easiest two-phase problem possible, that of a two-layered system in static equilibrium, particles will oscillate around the interface if the background grid is not aligned with the interface. This observation calls again for an accurate interface treatment in order to avoid the spurious oscillations. It can in fact be concluded that hybrid particle-mesh methods are in this respect more akin to Eulerian methods than Lagrangian methods such as SPH, because the same problems as in Eulerian methods are observed around the free surface.

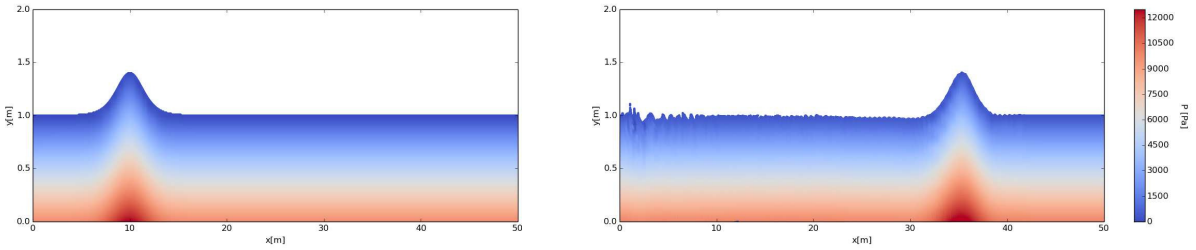


Figure 6.10: Pressures in propagating solitary wave for $H/d = 0.4$ at $t = 0$ s (left) and $t = 7$ s (right).

The simulated propagation of the solitary wave is compared to the analytical solution for different relative wave heights. Simulated surface elevations are extracted using a simple level set, see Appendix C. Simulated results are in good agreement with the theory as both the shape and the height of the solitary wave are well preserved over time. A phase lag of computed results with respect to the analytical solution is observed for all relative wave heights, but it appears to be most pronounced for the highest relative wave heights. This lagging behavior of the scheme was also observed in the standing wave test case, and was primarily attributed to the numerical diffusion in the scheme. It is however noted that for the solitary wave test case the accuracy of the analytical expression may also partly explain discrepancies between simulated results and analytical values, especially for large relative wave heights.

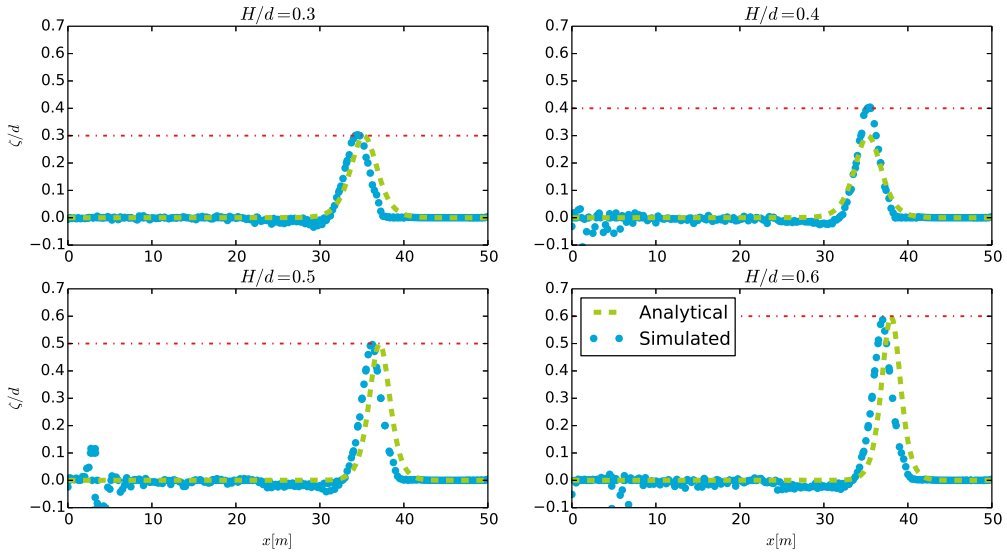


Figure 6.11: Solitary wave propagation: analytical versus simulated surface elevation at $t = 7\,\mathrm{s}$ for different relative wave heights.

6.4 Wavemaker boundary

In order to test the ability of the method for setting-up a numerical wave flume, the implementation of a wave generating boundary condition is assessed. Since such boundary conditions are an essential step in order to obtain a numerical wave flume, the topic has extensively been studied in literature. Without giving an in-depth overview of literature, three different approaches in forced wave generation can be mentioned:

1. Wave generation by means of mass source functions, where an additional term is added to the mass balance (see e.g. [75]).
2. A second approach is the use of open boundaries, on these boundaries a surface elevation and velocity are imposed which are derived from the relevant wave theory (see e.g. [76]).
3. The third approach is to make use of numerical wave makers. The fluid is thus forced by a moving object, having a prescribed motion. The moving object can be a rigid body, moving on the background grid (see e.g. [15]), or alternatively using a moving background grid and imposing the proper velocity boundary condition at the background grid.

Especially the latter method is popular in the field of Lagrangian methods such as SPH. Reason for this is obvious: on the one hand it is easy to give boundary particles (SPH) or the boundary vertices (hybrid methods) a prescribed motion, on the other hand it is difficult to implement a mass source function or an open boundary condition in the (semi-)Lagrangian methods since this would require the insertion of particles during the computation. Therefore, in this thesis on hybrid methods, a moving boundary will be used in order to generate waves. It would be a study in itself to develop a wave generator for various wave theories. Therefore, this thesis will mainly focus on the conceptual implementation of a wavemaker boundary, where the discussion is only limited to solitary wave generation. Thus, it should be kept in mind that one important aspect of wave generation is omitted, namely the wave absorption of reflected waves by the wavemaker. This part is crucially important for simulating near-realistic wave fields in a flume, as wave re-reflection will lead to simulated wave fields nowhere close to the physical solution.
It is immediately recognized that the implementation of the normal velocity boundary condition as outlined in Section 4.3.3 represents a numerical (piston-type) wave maker, and so the piston motion follows from known wave generation theories. In this approach the earlier mentioned feature of hybrid methods is utilized in that the background mesh can be redefined arbitrarily between consecutive timesteps.
However, in the developed two-phase approach,the top boundary requires special attention as the domain is entirely filled with water and air particles. An open boundary implementation as presented in Section 4.3 (grid related part) and Section 4.5 (particle related part) is used along the top boundary.

6.4.1. Numerical test case 1: solitary wave generation

As a test case for the moving boundary, a solitary wave is generated in the domain initially defined by $\Omega = [0, 15] \times [0, 0.6]$m. Theory behind the forced generation of solitary waves by means of piston wave makers can be found in [77] and [78]. Based on these references the displacement and velocity of the moving boundary is prescribed as:

$$x_b(t) = S \tanh\left(7.6\left(\frac{t}{\tau} - 0.5\right)\right) + S \tag{6.21}$$

$$\mathbf{u}_D(t) \cdot \mathbf{n} = S\frac{7.6}{\tau}\text{sech}^2\left(7.6\left(\frac{t}{\tau} - 0.5\right)\right) \tag{6.22}$$

In these equations, τ is a characteristic time duration for the paddle motion, and S the paddle stroke, respectively given by:

$$\tau = \frac{4}{\beta c}\left(\text{artctanh}(0.999) + \frac{H}{d}\right) \tag{6.23}$$

$$S = 2\sqrt{\frac{Hd}{3}} \tag{6.24}$$

with the wave celerity again given by $c = \sqrt{g\,(d + H)}$ and H and d are the wave height and the undisturbed water depth. Furthermore β is the so-called outskirts decay coefficient, defined as [77]:

$$\frac{\beta}{2} = \sqrt{\frac{3H}{4d^3}} \tag{6.25}$$

Table 6.4: Model properties solitary wave generation.

Model parameters	Ω_0	$[0, 15] \times [0, 0.6]$m	Computational domain
	nx×ny	500×20	Cells in x- and y-direction (diagonal)
	Δx_p	0.005m	Particle spacing
	α	0.03	PIC fraction
	Δt	$1E-3$ s	Time step
	d	0.3 m	Still water depth
	ν	0 m^2s^{-1}	Kinematic viscosity
Computational aspects		360000	Number of particles
		18	Average nr. particles per element
		10 s	Simulated physical time
		176 min	Model run time

Model settings for the numerical experiment are listed in Table 6.4. The model is run for two different dimensionless amplitudes, $\frac{H}{d} = 0.2$ and $\frac{H}{d} = 0.35$. Details about the moving boundary implementation in FEniCS can be found in Appendix B. The simulated air-water surface is again extracted using the level set approach described in Appendix C.

The relative surface elevations $\frac{\zeta}{d}$ for the $\frac{H}{d} = 0.2$ test case are depicted in Figure 6.12. The characteristic timescale τ can be conveniently used to fit the analytical solution onto the simulated solution [1]. Computed results are in good agreement with the analytical solution, as the amplitude and shape of the pulse are well preserved over time. A small trailing wave is observed behind the main pulse for both simulations, which can be expected based on physical experiments ([77], [78]) as well as numerical experiments [79]. In line with the standing wave test and the solitary wave propagation benchmark test, the phase speed of the simulated wave appears to be somewhat smaller than the analytical phase speed.

The relative surface elevations for the $\frac{H}{d} = 0.35$ testcase are given in Figure 6.13. Again, the generated soliton does correspond well with theory, although the relative amplitude of the generated wave is somewhat overestimated and the amplitude is damped over time. Possible explanation for this is that the wave

[1] The simulated solution lags a time $\frac{\tau}{2}$ behind the analytical solution.

generation theory as defined by Eqs. (6.21)-(6.22) is only accurate for relatively small waves (with $\frac{H}{d} \leq 0.2$) [80].

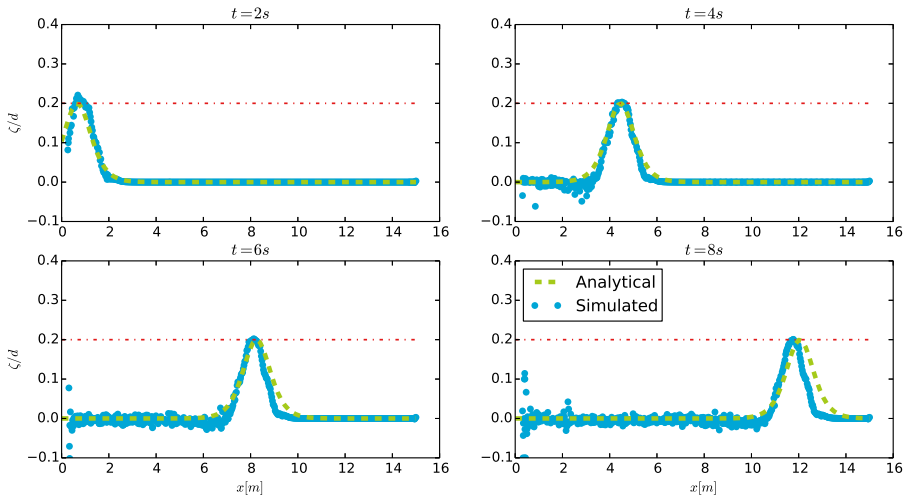


Figure 6.12: Relative surface elevation for solitary waves generated with the moving boundary compared to analytical solutions. Relative amplitude $\frac{H}{d} = 0.2$.

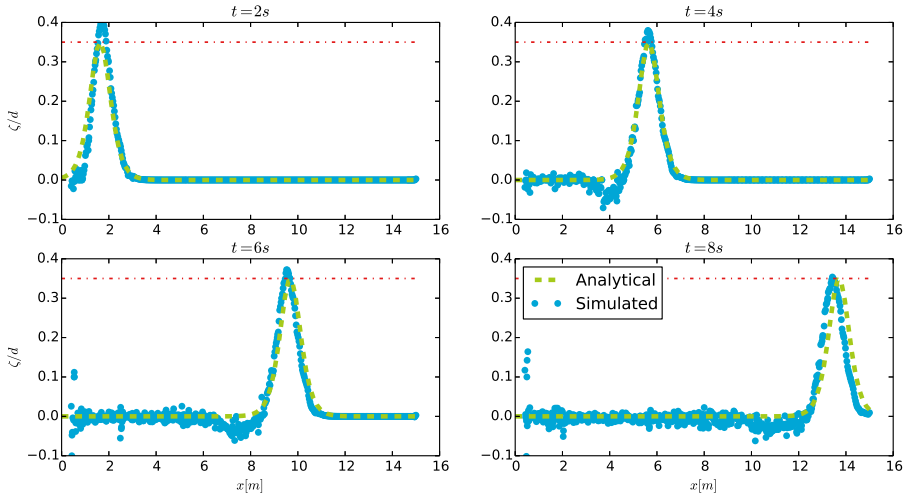


Figure 6.13: Relative surface elevation for solitary waves generated with the moving boundary compared to analytical solutions. Relative amplitude $\frac{H}{d} = 0.35$.

Given the above results, it can be concluded that hybrid particle-mesh methods are perfectly suitable for the implementation of moving boundaries. Although only a solitary wave generator was presented in this section, the extension to more sophisticated wave generation theories should be possible and relatively straightforward. Again, the problems near the air-water interface are clearly observed in the simulations. As can be seen in Figure 6.12 and Figure 6.13 the free surface trailing the main pulse does not come to rest, but instead a persisting oscillation around the still water level is observed. This phenomenon was also observed and explained in the previous sections and indicates that an accurate free surface treatment is necessary as the 'free surface wiggles' may render the computation to instability in a later stage of the computation.

Comparison with SPH

At this point it is perfecty feasible to make a comparison between the developed model and an existing SPH code in terms of computational effort and accuracy. To this end, the open-source SPH code DualSPHysics [81] is used to redo the benchmark described above. The DualSPHysics package is a rather mature SPH package and has been extensively used for academic research (an overview of publications can be found at `dual.sphysics.org`). DualSPHysics offers the option to solve the governing equations (for which reference is made to [3]) on the central processing unit (CPU) or on the graphics processing unit (GPU). Arbitrary geometries can be either defined inside DualSPHysics or imported from external (CAD) files. Furthermore a predefined motion can be given to objects inside the domain. In order to set-up a numerical wave tank for generating a solitary wave, the motion of the left boundary was prescribed according to Eq. (6.21).

In order to keep the comparison between the formulated model and the DualSPHysics implementation as honest as possible, the CPU implementation (single-thread) of DualSPHysics was initially used since all computations performed with the developed model were run on a single core. Furthermore, in analogy to Table 6.4 a particle spacing of 0.005 m was used, resulting in a total number of 186 798 particles (which is approximately half the amount of particles present in the hybrid particle-mesh computation, since only the water phase is considered). The timestep is automatically calculated by setting the CFL condition to 0.2, resulting in a typical time step of $O(5E-5\,\mathrm{s})$. Finally, for the problem under consideration the artificial viscosity, used for parameterization of the viscous term or simply enhancing the numerical stability [82], is set to 0. In other words, the flow is assumed to be inviscid.

Running the computation on the CPU using one core resulted in a model runtime of approximately 500 min, compared to the model run time for the developed hybrid particle-mesh method of approximately 180 min, Figure 6.14. Running the SPH model in parallel, using 4 cores, resulted in a model run time of approximately 180 min, hence being of the same order as the runtime for the developed model, where it is again remarked that the developed model was run on one core only, using even the double amount of particles. This analysis thus shows that the hybrid particle-mesh mehtod is computationally by far less expensive than the full particle methods such as SPH. In order to accelerate the SPH compuations, the GPU implementation of DualSPHysics can be used. Running the model with this setting and using an NVIDIA Quadro K2200 GPU, the model run time is reduced to approximately 30 min.

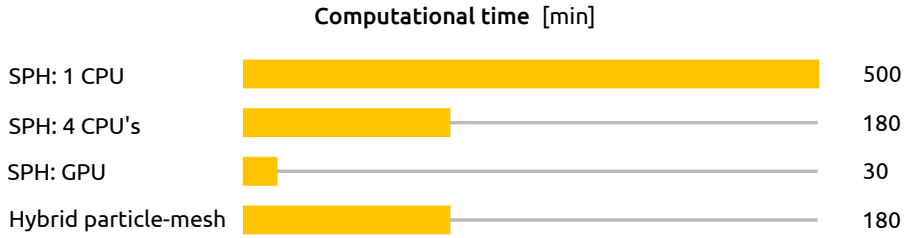


Figure 6.14: Approximate computational times for the SPH model in various computing configuration and the developed hybrid particle-mesh method.

The results obtained with the developed hybrid particle-mesh method and the DualSPHysics model are compared at time instance $t = 8\,\mathrm{s}$, both for a relative wave height of the soliton of $\frac{H}{d} = 0.2$ and $\frac{H}{d} = 0.35$. The results are presented in Figure 6.15. Compared to the developed model, the amplitude of the soliton is much more damped in the SPH simulation. This points to the fact that the numerical diffusion is much more pronounced in the SPH model, even in the absence of any artificial viscosity, compared to the developed particle-mesh method. Inspection reveals that the damping in the SPH method quickly increases when increasing the artificial viscosity parameter.

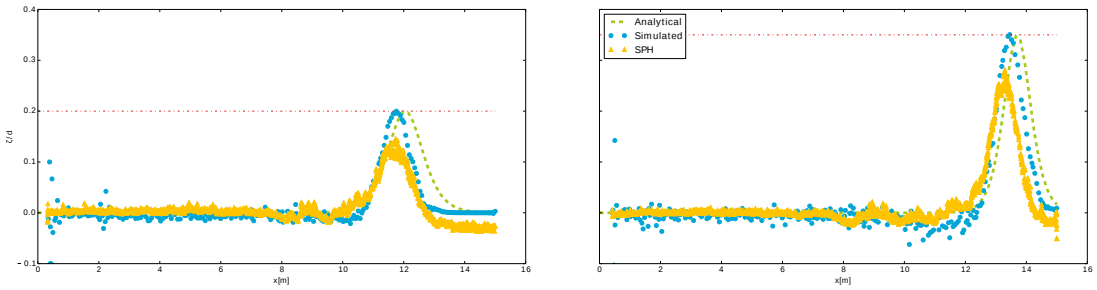


Figure 6.15: Comparison between the analytical, the hybrid particle-mesh and the SPH solution for the generated solitary wave at $t = 8\,\mathrm{s}$. Left panel corresponds to the $H/d = 0.2$ case, the right panel corresponds to the $H/d = 0.35$ case.

Based on the above presented comparison between the developed hybrid particle-mesh method and the SPH method, it can be concluded that the results obtained with the hybrid particle-mesh method are encouraging, both in terms of computational time and in terms of accuracy, since the numerical diffusion in the developed model is significantly lower compared to the SPH model.

6.4.2. Numerical test case 2: breaking wave

As a final example, showing the capabilities of the model, the wave breaking over a submerged bar is considered. To this end, computations are run on an irregular mesh generated with GMSH. Model parameters are listed in Table 6.5. Furthermore, solitary wave generation theory as presented in Section 6.4.1 is used to generate a solitary wave with relative amplitude $\frac{H}{d} = 0.625$. It is noted that a relatively high PIC fraction of $\alpha = 0.05$ was used in order to obtain stable model runs.

Table 6.5: Model properties breaking wave simulation.

Model parameters	Ω_0	$[0, 13] \times [0, 1]$m	Bounding box computational domain
	Δx_p	0.005m	Particle spacing
	α	0.05	PIC fraction
	Δt	$1E-3\,\mathrm{s}$	Time step
	d	$0.4\,\mathrm{m}$	Still water depth
	ν	$0\,\mathrm{m^2 s^{-1}}$	Kinematic viscosity
Computational aspects		24561	Number of cells in GMSH grid
		460 000	Number of particles
		19	Average nr. particles per element
		$6\,\mathrm{s}$	Simulated physical time
		156 min	Model run time

In Figure 6.16 the process of wave propagation and wave breaking is presented for different time instances. Different processes are clearly observed in this figure. First of all, in between $t = 3.0\,\mathrm{s}$ and $t = 4.5\,\mathrm{s}$ the decreasing water depth results in wave shoaling. This shoaling process results in a steepening of the wave front and a flattening of trailing edge of the wave. Ultimately, this process of increasing wave steepness results in wave breaking, starting to take place in between $t = 4.5\,\mathrm{s}$ and $t = 5\,\mathrm{s}$. At $t = 5\,\mathrm{s}$ the profile of a plunging breaker is observed, where the wave crest curls over and encloses a pocket of air. The plunging breaker impinges as a jet on the still water in front of the wave, resulting in a secondary splash-up after wave breaking ($t = 5.5\,\mathrm{s}$). Although not analyzed quantitatively, the model shows the expected behavior, revealing the capability of hybrid particle mesh methods to simulate flows having complex free surface topologies.

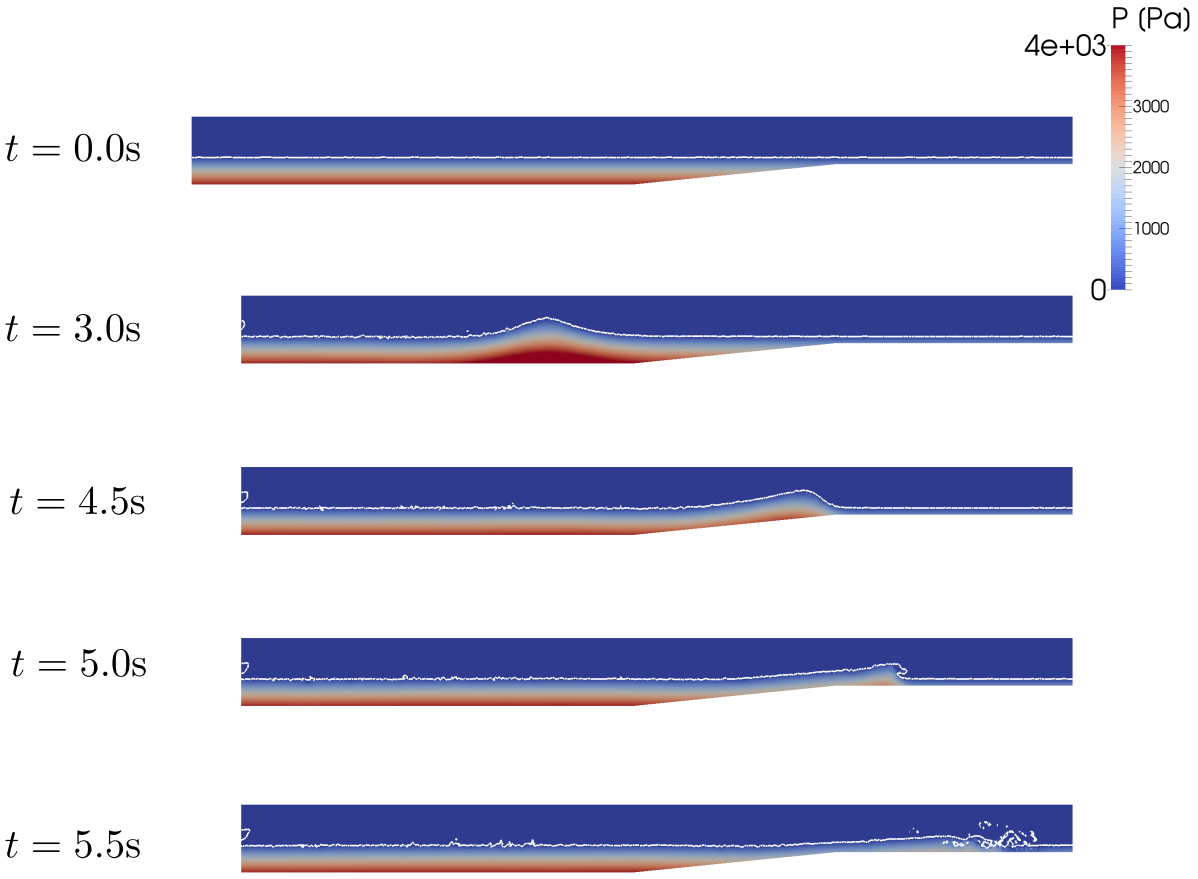


Figure 6.16: Soliton propagation including breaking on a submerged bar. Color shading indicates the pressures at the background grid.

6.5 Summary and conclusions

In this chapter several test cases were run for two-phase fluid simulations. The unmodified scheme turned out to work well for the small density difference Rayleigh-Taylor test case, in which the density of the two-phases were of the same order of magnitude. Running the standing wave simulation with the water-air density ratio of 1000, spurious velocities around the interface spoiled the model runs. This artifact can be attributed to the density jump near the interface. In order to mitigate this problem, it was recognized that although the pressure gradient and the density are discontinuous along the interface, the term $\frac{1}{\rho}\nabla p$ is continuous irrespective of the phase (provided that surface tension is neglected). Elaborating upon this notion, resulted in an additional mapping of the pressure gradient term. A step which was shown to *alleviate* the pathological spurious velocities near the interface, although it was also noted that this step is not a fundamental solution to the interface problem. Therefore, the list of 'weak spots' in the model formulation as presented in Section 4.7 is extended with the following item:

- In order to deal with two-phase materials, having large density differences, an additional mapping of the pressure gradient term is required. Although effectively oppressing the spurious interface velocities, a more fundamental solution to this problem is desirable. It is expected that techniques from multi-phase grid-based methods can be used in order to solve this problem. Examples of such approaches are the XFEM method or diffuse interface methods.

From the standing wave test it followed that although a sharp interface is maintained during the computation, the quality of the interface slowly degrades. This was further confirmed by the solitary wave propagation test, where oscillations around the interface were observed in the trailing part of the wave. This was attributed to the fact that static equilibrium can neither be obtained at element level nor at particle level for arbitrary interface locations. This artifact also leads to some residual particle settling,

resulting in a decay of potential energy over time.
In line with the test cases from Chapter 5, a significant amount of numerical diffusion is present in the system resulting in an energy decay as observed in the Rayleigh-Taylor test and the standing wave test. However, qualitative comparison to an SPH method for the standing wave test revealed this energy decay in the formulated method to be smaller than in the reference SPH method.

The implementation of a non-homogeneous kinematic boundary condition was assessed by formulating a wave generating boundary condition. Using a combination of a kinematic boundary condition at the side wall (similar to a piston-type wavemaker) and an open boundary condition at the top of the domain, a solitary wave was generated. Simulated results are in very good agreement with the analytical results, showing one of the strengths of the hybrid particle-mesh methods to deal with moving boundaries. On top of that the results were shown to improve upon an existing SPH implementation, both in terms of computational time and accuracy.
As a final example, wave breaking over a submerged bar was simulated. Qualitatively, the nearshore wave propagation processes are present in the model simulations, including wave shoaling, wave breaking and the post-breaking stage. One of the advantages of the hybrid particle-mesh method compared to Eulerian methods, is that the complex topology of the free surface is well-represented.

In short, it is concluded that the hybrid particle mesh implementation is well-suited for setting-up a numerical wave flume. Distinct advantages of the method compared to both Eulerian and Lagrangian methods is the straightforward implementation of the kinematic boundary condition. On top of that, the method is computationally less expensive and shows smaller numerical diffusivity compared to for example SPH. Furthermore, the incompressibility constraint can be conveniently enforced at the background grid. As a result, the time step in the developed method is $O(10^2)$ larger compared to weakly compressible MPM or SPH formulations used for the same applications. A distinct advantage of the hybrid implementation over the Eulerian methods is the representation of free surfaces having complex topological shape.
Despite the mentioned advantages of the method, it remains to be noted that a more fundamental solution for the interface treatment is desirable.

7

Conclusions and recommendations

In this thesis a hybrid particle-mesh method was developed for the simulation of free surface flows. More specifically the aim of this study was to assess whether hybrid methods are feasible and favourable for setting-up a numerical wave tank. This chapter will give an answer to the main question by recapping the conclusions to the three sub-questions from which an answer to the main question is formulated. Based on the research outcomes, an outlook for future research is given by formulating some recommendations.

7.1 Conclusions

In analogy to the research questions formulated in Section 1.3, the conclusions are sub-divided in three categories:

1. Conclusions regarding the hybrid particle-mesh methods.
2. Conclusions regarding the model formulation and model implementation.
3. Conclusions regarding the model results.

Conclusions regarding the first research question can be found in Section 2.5, conclusions regarding the second research question can be found in Section 3.5 and Section 4.7 and reference to Section 5.3 and Section 6.5 is made for conclusions regarding the model results.
Based on these conclusions, the main research question can be answered. This question was formulated as:

What are the prospects of hybrid particle-mesh methods in simulating free surface flows and is it possible to set-up a numerical wave flume using a hybrid method?

This main question is answered by a listing of the main advantages and main disadvantages of the method which were identified during the research.

The main advantages of hybrid particle-mesh methods can be identified to be the following:

- Contrary to Eulerian based methods, it is easy to deal with multi-material problems where the material interface is of complex topological shape (such as in wave breaking).
- Contrary to Lagrangian methods, it is straightforward to assume the flow to be strictly incompressible. By choosing an admissible element, the advantage of this assumption is two-fold: 1) the resulting pressures are accurately computed and 2) the timestep restriction for explicit methods becomes less stringent. The latter contributes to the fact that the developed scheme is computationally far less expensive than e.g. SPH.
- Hybrid particle mesh methods offer a huge flexibility in the interaction between grid and particles. Depending on the model formulation, responsibilities can be 'shifted' between particles and grid.
- Implementation of kinematic boundary conditions is straightforward. This feature was shown to be advantageous for setting-up a numerical wave flume.

Disadvantages of the hybrid methods are identified to be the following:

- It is far from understood what the optimal choice is for the emphasis on either the grid or the particles. For example, in the chosen model implementation an optimal approximation of the integrals (in the grid-sense) comes at the expense of loosing conservation of physical quantities during the particle-grid interaction step and a lack of control over the particle distribution. Putting more responsibilities on particle level will unavoidably lead to more noise in the results.
- Fundamental issue in hybrid methods is that the number of particle dofs exceeds the number of dofs which are solved (i.e. the background grid dofs). As a result, noise can develop at particle level. Methods to oppress this noise are somewhat ad hoc (e.g. the blending parameter for a PIC-FLIP velocity update).
- Specific to the present implementation, it is to be remarked that different 'questionable' steps were taken:
  - The particle-to-grid mapping, loosing control over the particle distribution.
  - A continuous velocity field was constructed in order to advect the particles.
  - Particle properties were updated using a blended PIC-FLIP approach. The value of the parameter controlling this step was chosen somewhat ad hoc.
  - An additional mapping of the pressure gradient term was required in order to oppress the spurious velocities at the air-water interface.

  Higher confidence in the hybrid particle-mesh methods can only be obtained by either avoiding these steps or reformulating them in a physical and mathematical correct way (see recommendations).

In short it can be said that hybrid particle-mesh methods appear to be an attractive tool for simulating free surface flows. Nevertheless, many fundamental questions remain unanswered when considering hybrid methods. It is to be remarked that all these questions can be basically reduced to the question how to interpret the interaction between particles and grid.

7.2 Recommendations

Recommendations for future research logically follow from the above presented conclusions. Three categories of recommendations are distinguished: recommendations regarding the model fundamentals, recommendations regarding the model formulation, and practical recommendations.

7.2.1. Recommendations regarding the model fundamentals

- There is still a lack of knowledge regarding the interaction between the particles and the grid. To end-up with a reliable hybrid method, it is important to develop a coherent and consistent vision on this interaction process. In other words how to physically and mathematically interpret the relation between particles and grid. Developing such a vision requires to take a step back and consider simple 1D 'toy problems' and focus on the mapping process between particles and grid only. As a starting point for this research, four different directions can be identified:
  - Considering the grid points as sample points of the continuum. This typically leads to least square approaches where particle values are used in order to construct a background grid field. See e.g. [50] and [51].
  - Considering the grid as a representation of the continuum, and the particles as a means of advecting the continuum quantities. This typically lead to classical particle-in-cell approaches, which are still extensively used in plasma physics. See e.g. [83] and [84].
  - Considering the particles as a representation of the continuum. This typically lead to a MPM approach, for which numerous references have been given in the text.
  - Considering the grid and the particles to be complimentary to each other. In other words, the particles and the grid live besides each other, but share information for certain computational steps. As such, the developed approach can be somewhat regarded as a first step in this direction since it has been stressed several times that in the developed approach the particle representation of physical quantities does not necessarily equal the grid representation of the physical quantities.

Complicating factor is that the method should be able to deal with strong discontinuities at the material interfaces.
- Fundamental issues were observed at the air-water interface. Future research should focus on an accurate interface description. As possible research directions, the XFEM method [70] or diffuse interface methods such as the phase field model approach [71] can be mentioned.

7.2.2. Recommendations regarding model formulation

Various recommendations can be given regarding the model formulation itself:

- In the present formulation, particles are advected in the grid velocity field. As an alternative to this, advection with the particle velocity itself can be tested. This has the advantage that it is not required anymore to construct a continuous velocity field at the background grid, thus pleading in favour of the P_0P_1element, or other discontinuous elements such as the $P_{-1}P_2$ element
- Related to the previous recommendation: one of the doubtful steps in the present implementation is the mapping of the P_0 velocities to P_1^+ velocities which was required to obtain a continuous velocity field for the particle advection. As a solution to this, other element types should be considered. Most obvious choice would be to use bubble elements (such as the $P_1^+P_1$element) or the P_2P_1 element.
- Not much attention was paid on how to enforce boundary conditions at particle level in this report. The open boundary condition was for example implemented in a somewhat ad hoc fashion. Future research should focus on a more accurate treatment of (open) boundaries, SPH literature on this topic can be helpful in this (see e.g. [85])

7.2.3. Recommendations regarding model implementation

Practical recommendations mainly concern the model development itself. To name a few important topics which should be considered in the future:

- The particle administration is far from being trivial. Optimization at this point is possible. This will especially involve fast and accurate particle tracking algorithms (for example required for the generalization to 3D). The availability of high-end particle libraries such as Aboria or Partio may become useful in a later stage to achieve this.
- In the present model implementation FEniCS was used as the FEM package. This choice should be critically reviewed. Indeed, FEniCS offers a huge variety of options when it comes to 'regular' FEM problems, however, at the moment it is not suited for flexibly dealing with particles. It is for example not possible to perform the quadrature at particle level, in other words, in the present version of FEniCS it is not possible to discretize problems with an MPM approach.

A

Solving variational forms in FEniCS

In this thesis, the open source package FEniCS is used for solving the governing equations at the background grid. The FEniCS project aims at automating the solution of partial differential equations (PDEs) by providing a generic framework for formulating, discretizing and solving variational forms. The philosophy behind the FEniCS project is to generate low-level, efficient code from a high-level, nearly mathematical formulation of the problem. by using this generic approach for the problem formulation, FEniCS has been applied to a vast range of engineering fields, involving among others hemodynamic problems, fluid-structure interaction prolems, and solid mechanics problems [59].
Without being complete, this appendix provides a brief overview of the core components of the FEniCS package. For a more extensive overview, reference is made to [86] and for an in-depth overview, reference is made to [59].

Solving PDEs using the FEM discretization technique, falls apart in four main steps [86]:

1. Formulate the variational form of the PDEs
2. Discretize the variational forms
3. Finite element assembly
4. Solve the system of equations

In FEniCS , the above mentioned steps are performed by different software components, enhancing the modularity of the package. These components together form the solution chain for solving PDEs using the FEM discretization. The FEniCS chain for the implementation of these steps is depicted in Figure A.1, although it is noted that there is not a one-to-one correspondence between the four FEM discretization steps and the four main components as presented in the figure. Therefore, a very brief description of the FEniCS main components is given below.

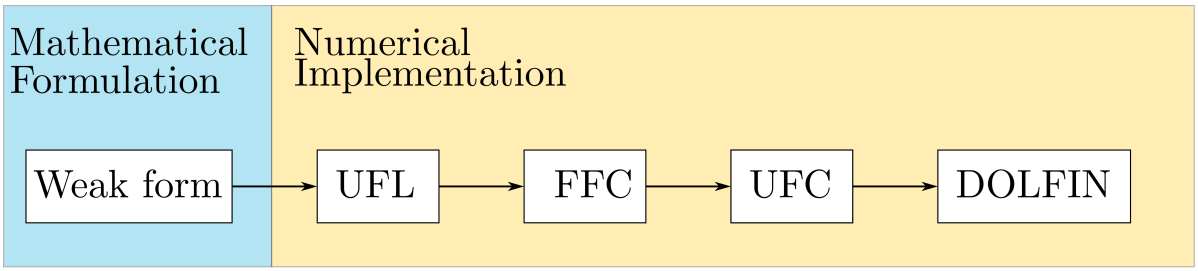


Figure A.1: The FEniCS toolchain for solving PDEs using FEM.

Unified Form Language

Any numerical computation will start with the problem definition, and more precisely a *discrete* formulation of the problem. Applied to the FEM method, this means that the variational form of the governing equations has to be derived and the finite dimensional solution spaces have to be defined.
The Unified Form Language (UFL) defines the language in which to *express* the governing equations for the problem under consideration. It does so by offering tools for the definition of the finite element, the variational form and mathematical expressions (i.e. algebraic functions).
In the UFL language, the reference *finite element* can be constructed using the following syntax:

```
element = FiniteElement(family, cell, degree)
```

Whereas a whole range of finite element families are supported in FEniCS , in this thesis only the "Lagrange" elements (alternatively denoted as "CG") and the "Discontinuous Lagrange" elements (alternatively denoted as "DG") are used. The ‘cell’ argument denotes the polygonal shape of the element, in this thesis only ‘triangle’ elements are used. Finally, the ‘degree’ argument specifies the order of the order of the polygonal shape functions.
When interfaced by DOLFIN (to be discussed next), the following UFL syntax is used when defining the *finite dimensional function spaces*:

```
P = FunctionSpace(mesh,family,degree)
```

Compared to the UFL code for the single *finite element*, the DOLFIN FunctionSpace class takes the 'mesh' variable as an argument. This variable contains the computational mesh.
The above code snippet presents the definition of a scalar function space, the UFL language does however also provide tools for constructing vector and tensor function spaces, both used in the thesis to respectively define the finite dimensional function spaces for the velocity and the viscous stress:

```
V   =   VectorFunctionSpace(mesh,family,degree,dimension)
Tau =   TensorFunctionSpace(mesh, family, degree)
```

When defining an instance of the VectorFunctionSpace class, the optional 'dimension' argument can be used to define the number of components of the vector. This argument defaults to the dimension of the given cell, i.e. in $\mathbb{R}^d$ the default dimension is d.
After definition of the function spaces, the trial and test functions can be defined. Sticking to the scalar function space P defined above, trial and test functions are respectively defined in UFL as:

```
p = TrialFunction(P)
q = TestFunction(P)
```

One of the key strengths of the UFL language is that it provide a set of tools for the near-mathematical expression of the variational form arising from the FEM discretization of the governing equations. As such, an overview of the syntax for various algebraic and differential operators often used in this thesis is presented in Table A.1.

Table A.1: Table of tensor algebraic operators (left). Table of differential operators (right). Only operators which are used in this thesis are shown.

Mathematical notation	UFL notation
$A \cdot B$	dot(A,B)
$A : B$	inner(A,B)
A^T	A.T

Mathematical notation	UFL notation
∇A	grad(A)
$\nabla \cdot A$	div(A)

In the main report various examples are presented how this syntax can be used for the representation of variational forms. One important thing to note is the representation of domain and boundary integrals in the UFL language: domain integrals of the form $\int_\Omega I \mathrm{d}\Omega$ are expressed in the form I*dx(k) and boundary integrals of the form $\oint_{\partial\Omega} I \mathrm{d}\Gamma$ are denoted by I*ds(k) where k is the subdomain number and I can be any valid UFL expression.

FEniCS Form Compiler

Having the discrete variational problem expressed in the UFL representation, the FEniCS Form Compiler (FFC) is invoked. Main purpose of the FFC is to turn the high-level, near-mathematical problem description (UFL) into low-level C++ code which can be interpreted in the subsequent steps (UFC/DOLFIN). As such, FFC generates the code which can be used in subsequent steps for evaluating the element tensors and the local-to-global mapping from the reference element to the elements in the computational mesh.
Important to note is that FFC does not do any computational work, instead, it only generates *problem-specific* code for the evaluation of the element tensor and the local-to-global mapping which can be called in a later stage by a general purpose routine in order to assemble the global system of equations.

Unified Form-assembly Code

The Unified Form-assembly Code (UFC) provides an interface between the problem-specific code generated by FFC and general-purpose problem solving environments like DOLFIN (to be described next). As such, the main purpose of UFC is to separate the implementation of the form and other details necessary for the assembly, such as mesh properties [87]. Discussion of the details of UFC are outside the scope of this thesis.

DOLFIN

The aforementioned FEniCS components are interfaced by the DOLFIN library. As such, DOLFIN is the main interface for expressing variational forms in the UFL language and controlling the compilation, assembling and solution step. Thus, the DOLFIN library wraps together the main FEniCS functionalities described above.

Apart from being the main interface, it is also one of the core components of the FEniCS package since the assembly of the global system and solving the system is done by DOLFIN. For these steps, and especially for solving the resulting system of equations, DOLFIN depends on linear algebra libraries such as PETSc and uBLAS [59].

The DOLFIN libraries on its turn can be interfaced by either C++ or Python. In this thesis, only the Python interface was used. Advantage over C++ is that use can be made of the expressiveness offered by Python when defining the problem and post-processing the results. Moreover, using Python as an interface also offers the flexibility to combine functionalities of the DOLFIN library with the FORTRAN based particle libraries.

B

Implementation of boundary conditions in FEniCS

This appendix presents the FEniCS implementation of various boundary conditions. The code snippets for both the moving mesh approach and the imposition of an essential and normal boundary condition are two simple working examples.

Periodic boundary condition

The periodic boundary condition is specified by constraining the function space domain. This is done by specifying a FEniCS SubDomain class for mapping the coordinates at complimentary boundaries. It is noted that this requires the boundary nodes at periodic boundaries to be aligned.

```
from dolfin import *

# Periodic boundary definition for the Taylor-Green test case
# Works only for the [-1,1]x[-1,1] domain

class PeriodicBoundary(SubDomain):
    # Left and bottom boundaries are "target domains"
    def inside(self, x, on_boundary):
        # return True IF on left OR bottom boundary
        # AND NOT on one (0, 1) OR (1, 0)
        return bool((near(x[0], -1) or near(x[1], -1)) and\
                (not ((near(x[0], -1) and near(x[1], 1)) or
                (near(x[0], 1) and near(x[1], -1)))) and on_boundary)
    def map(self, x, y):
        if near(x[0], 1) and near(x[1], 1):
            # Map top left corner
            y[0] = x[0] - 2.
            y[1] = x[1] - 2.
        elif near(x[0], 1):
            # Map left boundary
            y[0] = x[0] - 2.
            y[1] = x[1]
        else:
            # Then near(x[1], 1), map top
            y[0] = x[0]
            y[1] = x[1] - 2.

# Construct mesh
a       =       -1
b       =       1
c       =       -1
d       =       1
nx      =       30
ny      =       30
mesh  = RectangleMesh(Point(a,c),Point(b,d),nx,ny,'crossed')

# Define periodic CG1 scalar and vector spaces
CG1sP   = FunctionSpace(mesh,'Lagrange',1,
                        constrained_domain=PeriodicBoundary())
CG1vP   = VectorFunctionSpace(mesh,'Lagrange',1,dim=2,
                        constrained_domain=PeriodicBoundary())

# DG spaces definition, no changes
DG0s    = FunctionSpace(mesh,'DG',0)
DG0v    = VectorFunctionSpace(mesh,'DG',0,dim=2)
```

Moving mesh approach

The moving mesh approach is used for applying a wave generating boundary condition. The piece of code presented below was used for moving the mesh for the solitary wave generation test case, with a relative water depth $\frac{H}{d} = 0.35$ (see Section 6.4.1).

```
from dolfin import *
import numpy as np

# Python code for FEniCS moving mesh

def moving_bound_soliton(xn,xmax,ymax,to,t,g,h0,A):
    """ Function to prescribe boundary motion solitary wave:

    Requires the following arguments
    xn          ->      Old location of boundary
    xmax        ->      Moving domain definition
    ymax        ->      Moving domain definition
    to          ->      Old time level
    t           ->      New time level
    g           ->      Gravitational force
    h0          ->      Still water depth
    A           ->      Target amplitude

    Returns:
    u           ->      Boundary velocity
    xn          ->      New location of boundary
    And updated boundary.coordinates()
    """

    c        =      np.sqrt(g*(h0+A))
    beta     =      2*np.sqrt(0.75*A/h0**3)
    tau      =      4/(beta*c)*(3.8+A/h0)
    Sg       =      2*np.sqrt(A*h0/3)
    u        =      7.6/tau*Sg* (1/cosh(7.6*(t/tau-0.5)))**2
    dx       =      u*dt
    xn      +=      dx

    for x in boundary.coordinates():
        if between(x[0],(Sg*tanh(7.6*(to/tau-0.5))-0.001+Sg,
                         Sg*tanh(7.6*(to/tau-0.5))+0.001+Sg)):
            # Identify nodes located at the moving boundary
            x[0] = Sg*tanh(7.6*(t/tau-0.5))+Sg
        elif ( abs(x[1]) < 0.001 and x[0] < xmax ):
            # Identify nodes located at the bottom boundary,
            # move in order to keep good mesh quality
            x[0] +=  dx*(1/(xn-xmax)*(x[0]-xmax))
        elif ( abs(x[1]-ymax) < 0.001 and x[0] < xmax ):
            # Identify nodes located at the top boundary,
            # move in order to keep good mesh quality
            x[0] +=  dx*(1/(xn-xmax)*(x[0]-xmax))
    return u, xn


# Continued next page
```

```
# Continuation

# Domain parameters
xmax            =       15.
ymax            =       0.6

# Wave parameters
g               =       10.
h0              =       0.3
A               =       0.105
dt              =       0.001
tend            =       5.
t               =       0.
to              =       0.
xn              =       0

# Initialize mesh, define output file
mesh            =       RectangleMesh(Point(0,0),Point(xmax,ymax),300,20)
boundary        =       BoundaryMesh(mesh, "exterior")
meshFile        =       File("MovingTest/mesh.pvd")

while t<=tend:
        # Compute the boundary velocity uD, update the boundary coords
        uD,xn = moving_bound_soliton(xn,xmax,ymax,to,t,g,h0,A)
        # Use uD in order to enforce inhomogeneous normal velocity at

        # Move mesh, given updated boundary coords. Save in meshFile
        mesh.move(boundary)
        meshFile<<mesh
        to=t
        t+=dt
```

Imposing an essential and a natural boundary condition

Implementation of an essential and a natural boundary condition is shown for solving the Laplace problem on the unit square $\Omega = [0,1] \times [0,1]$ given by:

$$\nabla^2 p = 0 \qquad \text{in} \quad \Omega \tag{B.1}$$

$$p = \sin 2\pi x \qquad \text{on} \quad \Gamma_T \tag{B.2}$$

$$\nabla p \cdot \mathbf{n} = p_n = 10 \qquad \text{on} \quad \Gamma_L \tag{B.3}$$

with Γ_T denoting the top boundary and Γ_L the left boundary.
The corresponding variational form becomes: find $p_h \in \hat{\Omega}_h$ such that:

$$\int_\Omega \nabla p_h \cdot \nabla q_h \mathrm{d}\Omega = \oint_{\partial\Gamma_L} p_n q_h \mathrm{d}\Gamma \qquad \forall q_h \in \Omega_h \tag{B.4}$$

where the function space $\hat{\Omega}_h$ satisfies the inhomogeneous Dirichlet boundary condition and Ω_h satisfies the homogeneous boundary condition. The code below shows the implementation of the boundary condition in FEniCS .

```
from dolfin import *
import numpy as np

# Simple working example for specifying Dirichlet
# condition for pressure in FEniCS on the unitsquare

# Define open boundary class
class Top(SubDomain):
    def inside(self,x,on_boundary):
        return near(x[1],1.)

class Left(SubDomain):
    def inside(self,x,on_boundary):
        return near(x[0],1)

mesh = UnitSquareMesh(20,20)

# Function space and trial and test functions
CG1     =       FunctionSpace(mesh,"CG",1)
p       =       TrialFunction(CG1)
q       =       TestFunction(CG1)
gradp   =       Constant(10.)

# Natural bc at left boundary
bc0     =       Left()
bounds  =       FacetFunction("size_t", mesh)
bounds.set_all(0)
bc0.mark(bounds, 1)
ds      = Measure("ds")[bounds]

# Specify condition, p=0 at top boundary
bc1     =       Expression("sin(2*pi*x[0])",pi=np.pi)
TopBC   =       Top()
bcp     =       [DirichletBC(CG1,bc1,TopBC)]

# Variational form for Laplace equation
a       =       inner(grad(p),grad(q))*dx
l       =       -gradp*q*ds(1)

# Assemble and solve system
A       =       assemble(a)
L       =       assemble(l)
p       =       Function(CG1)
[bc.apply(A,L) for bc in bcp]
solve(A,p.vector(),L,"cg","ilu")
plot(p)
interactive()
```

C

A particle level set formulation

Level set functions are a popular method for locating and tracking the surface of a material or the interface between materials. A popular class of level set functions is the signed distance functions. Depending on the sign of this function, a point $\mathbf{x}$ is inside or outside the domain Ω:

$$\psi(\mathbf{x}) > 0 \qquad \text{for} \qquad \mathbf{x} \in \Omega \tag{C.1}$$
$$\psi(\mathbf{x}) < 0 \qquad \text{for} \qquad \mathbf{x} \notin \Omega \tag{C.2}$$

Consequently, if a point is located *on* the surface, the signed distance function takes a value of 0, in other words: the zero-isocontour of the signed distance function represents the surface of the domain Ω.
The concept of signed distance functions has become very popular in combination with two-phase, Eulerian fluid simulations such as Volume of Fluid. But also in the particle community, the value of the level set approach and specifically the signed distance function is recognized. Whereas in Eulerian methods, the signed distance function is an advected quantity itself (i.e. an additional advection equation is solved for the interface propagation), particle methods allow for a straightforward construction of the signed distance function from particle data.

In this thesis, such a signed distance function was introduced for post-processing purposes only. Therefore, not much effort is spent on the intricacies related to level sets in general and signed distance functions specifically (e.g. volume conservation). The approach employed in this thesis largely follow [6], in which a signed distance function was constructed in two-phase FLIP simulations.
The idea proposed in aforementioned paper is relatively straightforward: first a distance field is constructed for both the water phase and the air phase, respectively denoted with ψ_W and ψ_A. In analogy to [6], the value of these functions in the element barycenters is computed as:

$$\psi_\gamma(\mathbf{x}) = \min_{p \in P_\gamma} \parallel \mathbf{x} - \mathbf{x}_p \parallel \tag{C.3}$$

in which the subscript γ denotes either the air-phase A or the water phase W, and hence the set P_γ is defined as:

$$P_\gamma = \{p \,:\, \mathbf{x}_\mathrm{p} \in \Omega_\gamma\} \tag{C.4}$$

with Ω_γ either the water region Ω_W or the air region Ω_A.
By doing so, two additional values are stored at the element barycenter: the distance field for the water phase and the distance field for the air phase. These two distance functions can be merged into a *signed* distance function as:

$$\psi(\mathbf{x}) = \frac{\psi_W(\mathbf{x}) - \psi_A(\mathbf{x})}{2} \tag{C.5}$$

so that:

$$\psi(\mathbf{x}) > 0 \qquad \text{for} \qquad \mathbf{x} \in \Omega_A \tag{C.6}$$
$$\psi(\mathbf{x}) < 0 \qquad \text{for} \qquad \mathbf{x} \in \Omega_W \tag{C.7}$$

and $\Omega_A \cup \Omega_W = \Omega$ denote the air and the water domains respectively.
The discrete signed distance function is defined in the element barycenters. Since this can be expected to result in wiggly contours, the values at the barycenter are mapped to piecewise linears. Again it is emphasized that such a step is only justified by the fact that the level set formulation is used for post-processing purposes.
It must be noted that the construction of the distance function as specified by Eq. (C.3) is a computationally expensive operation, as it requires a nested loop over the particles inside looping over the elements. No further optimization of this procedure is considered since the level set approach in this thesis is only used for post-processing purposes and hence, the signed distance function is only computed at relatively large time intervals.
The performance of the level set approach is shown for a simple kinematic test case. Inside the unit square $\Omega = [0,1] \times [0,1]$ a single vortex is specified, with the velocity field specified as:

$$\mathbf{u}(\mathbf{x}) = 2 \begin{bmatrix} -\sin(\pi x)\sin(\pi y)\cos(\pi y) \\ \sin(\pi y)\sin(\pi x)\cos(\pi x) \end{bmatrix} \tag{C.8}$$

Initially, a disk of material A is centered at $(0.5, 0.75)$ with a radius of 0.15. This initially disk-shaped set of particles will stretch when subjected to the given velocity field. In Figure C.1 the particle representation and the corresponding level set is depicted. A 40×40 diagonal background grid is used. Other relevant parameters are listed in Table C.1.

Table C.1: Model settings level set test.

Ω_0	$[0, 1] \times [0, 1]$	Computational domain
nx×ny	40×40	Cells in x- and y-direction (diagonal)
Δx_p	0.003	Particle spacing
Δt	$1E-2\,\mathrm{s}$	Time step

The level set gives a proper representation of the interface, although it must be mentioned that the level set formulation has difficulties with representing very thin features (i.e. in a later stage when the disk is highly stretched). Nevertheless, it is concluded that the outlined approach can be used for post-processing purposes.

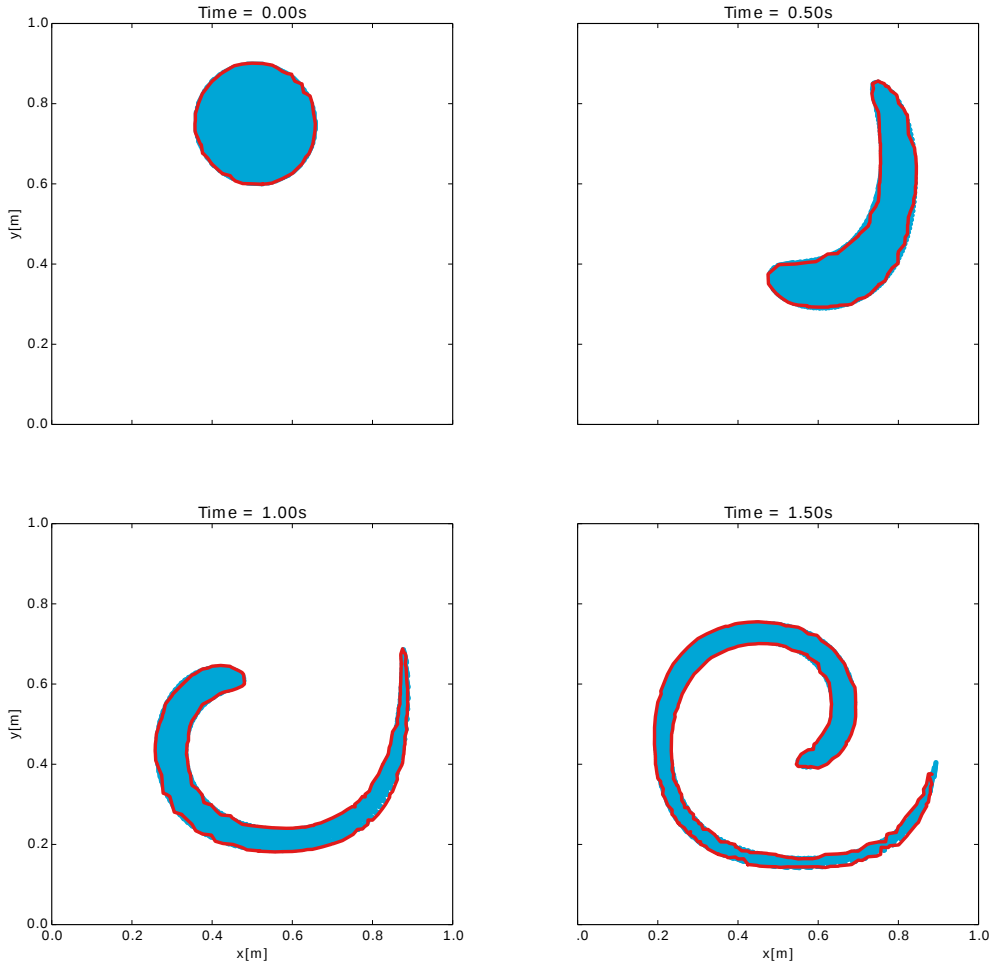


Figure C.1: Particle level set function in a single vortex flow field for different time instances.

Bibliography

[1] J. Maljaars, *On the applicability of numerical tools in the assessment of breakwater stability,* (2015).

[2] C. Hirt and B. Nichols, *Volume of fluid (VOF) method for the dynamics of free boundaries,* Journal of Computational Physics **39**, 201 (1981).

[3] J. Monaghan, *Simulating Free Surface Flows with SPH,* Journal of Computational Physics **110**, 399 (1994).

[4] C. Mast, P. Mackenzie-Helnwein, P. Arduino, G. Miller, and W. Shin, *Mitigating kinematic locking in the material point method,* Journal of Computational Physics **231**, 5351 (2012).

[5] F. M. Hamad, *Formulation of a Dynamic Material Point Method and Applications to Soil-Water-Geotextile Systems*, Ph.D. thesis, Universitat Stuttgart (2014).

[6] L. Boyd and R. Bridson, *MultiFLIP for energetic two-phase fluid simulation,* ACM Transactions on Graphics **31**, 1 (2012).

[7] J. Donea and A. Huerta, *Finite element methods for flow problems* (John Wiley & Sons, 2003).

[8] P. M. Gresho and R. L. Sani, *On pressure boundary conditions for the incompressible Navier-Stokes equations,* International Journal for Numerical Methods in Fluids **7**, 1111 (1987).

[9] G. R. Johnson, R. A. Stryk, and S. R. Beissel, *SPH for high velocity impact computations,* Computer Methods in Applied Mechanics and Engineering **139**, 347 (1996).

[10] S. Ma, X. Zhang, and X. Qiu, *Comparison study of MPM and SPH in modeling hypervelocity impact problems,* International Journal of Impact Engineering **36**, 272 (2009).

[11] A. Colagrossi and M. Landrini, *Numerical simulation of interfacial flows by smoothed particle hydrodynamics,* Journal of Computational Physics **191**, 448 (2003).

[12] W. Hu and Z. Chen, *Model-based simulation of the synergistic effects of blast and fragmentation on a concrete wall using the MPM,* International Journal of Impact Engineering **32**, 2066 (2006).

[13] D. Kothe, *Perspective on Eulerian Finite Volume Methods for Incompressible Interfacial Flows,* in *Free Surface Flows*, International Centre for Mechanical Sciences, Vol. 391, edited by H. Kuhlmann and H.-J. Rath (Springer Vienna, 1998) pp. 267–331.

[14] R. Xu, P. Stansby, and D. Laurence, *Accuracy and stability in incompressible SPH (ISPH) based on the projection method and a new approach,* Journal of Computational Physics **228**, 6703 (2009).

[15] D. M. Kelly, Q. Chen, and J. Zang, *PICIN:a particle-in-cell solver for incompressible free surface flows with two-way fluid-solid coupling,* Journal on Scientific Computing (2015).

[16] C. Jiang, C. Schroeder, A. Selle, J. Teran, and A. Stomakhin, *The affine particle-in-cell method,* ACM Transactions on Graphics **34**, 51:1 (2015).

[17] Y. Zhu and R. Bridson, *Animating sand as a fluid,* ACM Transactions on Graphics **24**, 965 (2005).

[18] R. Ando, N. Thürey, and C. Wojtan, *Highly adaptive liquid simulations on tetrahedral meshes,* ACM Transactions on Graphics **32**, 1 (2013).

[19] Z. Więckowski, *The material point method in large strain engineering problems,* Computer Methods in Applied Mechanics and Engineering **193**, 4417 (2004).

[20] S. Mao, Q. Chen, D. Li, and Z. Feng, *Modeling of Free Surface Flows Using Improved Material Point Method and Dynamic Adaptive Mesh Refinement,* Journal of Engineering Mechanics (2015).

[21] M. Evans, F. Harlow, and E. Bromberg, *The particle-in-cell method for hydrodynamic calculations*, Tech. Rep. (Los Alamos Scientific Laboratory, 1957).

[22] F. H. Harlow, *The particle-in-cell computing method for fluid dynamics,* Methods in computational physics **3**, 319 (1964).

[23] F. H. Harlow, *Fluid dynamics in Group T-3 Los Alamos National Laboratory,* Journal of Computational Physics **195**, 414 (2004).

[24] J. Brackbill, *The ringing instability in particle-in-cell calculations of low-speed flow,* Journal of Computational Physics **75**, 469 (1988).

[25] F. H. Harlow, *PIC and its progeny,* Computer Physics Communications **48**, 1 (1988).

[26] J. U. Brackbill, *Particle methods,* International Journal for Numerical Methods in Fluids **47**, 693 (2005).

[27] C. E. Gritton, *Ringing instabilities in particle methods,* (2014).

[28] C. Batty, F. Bertails, and R. Bridson, *A fast variational framework for accurate solid-fluid coupling,* ACM Transactions on Graphics **26**, 100 (2007).

[29] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle, *A material point method for snow simulation,* ACM Transactions on Graphics **32**, 1 (2013).

[30] J. Brackbill and H. Ruppel, *FLIP: A method for adaptively zoned, particle-in-cell calculations of fluid flows in two dimensions,* Journal of Computational Physics **65**, 314 (1986).

[31] J. Brackbill, *The ringing instability in particle-in-cell calculations of low-speed flow,* Journal of Computational Physics **75**, 469 (1988).

[32] R. Bridson, *Fluid simulation for computer graphics* (CRC Press, 2008).

[33] D. Sulsky, Z. Chen, and H. Schreyer, *A particle method for history-dependent materials,* Computer Methods in Applied Mechanics and Engineering **118**, 179 (1994).

[34] Z. Więckowski, S.-K. Youn, and J.-H. Yeon, *A particle-in-cell solution to the silo discharging problem,* International Journal for Numerical Methods in Engineering **45**, 1203 (1999).

[35] I. K. J. Al-Kafaji, *Formulation of a dynamic material point method (MPM) for geomechanical problems*, Ph.D. thesis, Universitat Stuttgart (2013).

[36] A. York, D. Sulsky, and H. Schreyer, *Fluid–membrane interaction based on the material point method,* International Journal for Numerical Methods in Engineering , 901 (2000).

[37] H. Lu, N. P. Daphalapurkar, B. Wang, S. Roy, and R. Komanduri, *Multiscale simulation from atomistic to continuum – coupling molecular dynamics (MD) with the material point method (MPM),* Philosophical Magazine (2007).

[38] S. Cummins and J. Brackbill, *An Implicit Particle-in-Cell Method for Granular Materials,* Journal of Computational Physics **180**, 506 (2002).

[39] S. Bardenhagen, J. Brackbill, and D. Sulsky, *The material-point method for granular materials,* Computer Methods in Applied Mechanics and Engineering **187**, 529 (2000).

[40] J. Bonet and A. J. Burton, *A simple average nodal pressure tetrahedral element for incompressible and nearly incompressible dynamic explicit applications,* Communications in Numerical Methods in Engineering **14**, 437 (1998).

[41] S. Andersen and L. Andersen, *Analysis of spatial interpolation in the material-point method,* Computers & Structures **88**, 506 (2010).

[42] M. Steffen, R. M. Kirby, and M. Berzins, *Analysis and reduction of quadrature errors in the material point method (MPM),* International Journal for Numerical Methods in Engineering **76**, 922 (2008).

[43] S. G. Bardenhagen and E. M. Kober, *The generalized interpolation material point method,* CMES - Computer Modeling in Engineering and Sciences **5**, 477 (2004).

[44] M. Steffen, P. Wallstedt, J. Guilkey, R. Kirby, and M. Berzins, *Examination and analysis of implementation choices within the Material Point Method (MPM),* CMES - Computer Modeling in Engineering and Sciences **31**, 107 (2008).

[45] R. Ando, N. Thürey, and R. Tsuruno, *Preserving fluid sheets with adaptively sampled anisotropic particles.* IEEE transactions on visualization and computer graphics **18**, 1202 (2012).

[46] D. Burgess, D. Sulsky, and J. Brackbill, *Mass matrix formulation of the FLIP particle-in-cell method,* Journal of Computational Physics **103**, 1 (1992).

[47] M. Thielmann, D. A. May, and B. J. P. Kaus, *Discretization Errors in the Hybrid Finite Element Particle-in-cell Method,* Pure and Applied Geophysics **171**, 2165 (2014).

[48] T. Belytschko, Y. Krongauz, D. Organ, M. Fleming, and P. Krysl, *Meshless methods: An overview and recent developments,* Computer Methods in Applied Mechanics and Engineering **139**, 3 (1996).

[49] T. Belytschko, Y. Y. Lu, and L. Gu, *Element-free Galerkin methods,* International Journal for Numerical Methods in Engineering **37**, 229 (1994).

[50] E. Edwards and R. Bridson, *A high-order accurate particle-in-cell method,* International Journal for Numerical Methods in Engineering **90**, 1073 (2012).

[51] P. C. Wallstedt and J. E. Guilkey, *A weighted least squares particle-in-cell method for solid mechanics,* International Journal for Numerical Methods in Engineering **85**, 1687 (2011).

[52] P. M. Gresho and R. L. Sani, *Incompressible flow and the finite element method* (John Wiley and Sons, Inc., New York, NY (United States), 1998).

[53] D. Arnold, F. Brezzi, and M. Fortin, *A stable finite element for the stokes equations,* CALCOLO **21**, 337 (1984).

[54] J. Leboeuf, T. Tajima, and J. Dawson, *A magnetohydrodynamic particle code for fluid simulation of plasmas,* Journal of Computational Physics **31**, 379 (1979).

[55] C. Geuzaine and J. F. Remacle, *GMSH: A 3-D finite element mesh generator with built-in pre- and post-processing facilities,* International Journal for Numerical Methods in Engineering **79**, 1309 (2009).

[56] Y. Deubelbeiss and B. Kaus, *Comparison of Eulerian and Lagrangian numerical techniques for the Stokes equations in the presence of strongly varying viscosity,* Physics of the Earth and Planetary Interiors **171**, 92 (2008).

[57] A. Quarteroni, F. Saleri, and A. Veneziani, *Factorization methods for the numerical approximation of Navier–Stokes equations,* Computer Methods in Applied Mechanics and Engineering **188**, 505 (2000).

[58] A. Ralston, *Runge-Kutta Methods with Minimum Error Bounds,* Mathematics of Computation **16**, 431 (1962).

[59] A. Logg, K.-A. Mardal, and G. N. Wells, *Automated Solution of Differential Equations by the Finite Element Method*, Vol. 84 (2012) p. 724.

[60] B. A. Wols, *CFD in drinking water treatment* (TU Delft, Delft University of Technology, 2010).

[61] S. Haber, N. Filipovic, M. Kojic, and A. Tsuda, *Dissipative particle dynamics simulation of flow generated by two rotating concentric cylinders: boundary conditions.* Physical review. E, Statistical, nonlinear, and soft matter physics **74** (2006), 10.1103/PhysRevE.74.046701.

[62] M. Kojic, N. Filipovic, and A. Tsuda, *A mesoscopic bridging scale method for fluids and coupling dissipative particle dynamics with continuum finite element method,* Computer Methods in Applied Mechanics and Engineering **197**, 821 (2008).

[63] L. J. Lim, *Pile Penetration Simulation with Material Point Method,* (2012).

[64] A. Colagrossi, B. Bouscasse, M. Antuono, and S. Marrone, *Particle packing algorithm for SPH schemes,* Computer Physics Communications **183**, 1641 (2012).

[65] L. de Wit, *Smoothed Particle Hydrodynamics. A Study of the possibilities of SPH in hydraulic engineering,* (2006).

[66] G. Tryggvason, *Numerical simulations of the Rayleigh-Taylor instability,* Journal of Computational Physics **75**, 253 (1988).

[67] X. He, S. Chen, and R. Zhang, *A Lattice Boltzmann Scheme for Incompressible Multiphase Flow and Its Application in Simulation of Rayleigh–Taylor Instability,* Journal of Computational Physics **152**, 642 (1999).

[68] Z. Więckowski, *Enhancement of the material point method for Fluid-Structure Interaction and Erosion,* (2013).

[69] R. Wemmenhove, *Numerical simulation of two-phase flow in offshore environments* (Citeseer, 2008).

[70] T. P. Fries and T. Belytschko, *The extended/generalized finite element method: an overview of the method and its applications,* International Journal for Numerical Methods in Engineering **84**, 253 (2010).

[71] D. Jacqmin, *Calculation of Two-Phase Navier–Stokes Flows Using Phase-Field Modeling,* Journal of Computational Physics **155**, 96 (1999).

[72] L. H. Holthuijsen, *Waves in oceanic and coastal waters* (Cambridge University Press, 2007).

[73] I. G. Main, *Vibrations and waves in physics* (Cambridge University Press, 1993).

[74] C. C. Mei, M. Stiassnie, and D. K.-P. Yue, *Theory and applications of ocean surface waves: nonlinear aspects*, Vol. 23 (World Scientific, 2005).

[75] P. Lin and P. L.-F. Liu, *Internal Wave-Maker for Navier-Stokes Equations Models,* Journal of Waterway, Port, Coastal, and Ocean Engineering **125**, 207 (1999).

[76] P. Troch and J. De Rouck, *An active wave generating–absorbing boundary condition for VOF type numerical model,* Coastal Engineering **38**, 223 (1999).

[77] D. G. Goring, *Tsunamis–the propagation of long waves onto a shelf*, Ph.D. thesis, California Institute of Technology (1978).

[78] G. Katell and B. Eric, *Accuracy of solitary wave generation by a piston wave maker,* Journal of Hydraulic Research **40**, 321 (2002).

[79] R. Labeur, *Finite element modelling of transport and non-hydrostatic flow in environmental fluid mechanics*, Ph.D. thesis, Delft University of Technology (2009).

[80] S. T. Grilli, M. A. Losada, and F. Martin, *Characteristics of Solitary Wave Breaking Induced by Breakwaters,* Journal of Waterway, Port, Coastal, and Ocean Engineering **120**, 74 (1994).

[81] A. Crespo, J. Domínguez, B. Rogers, M. Gómez-Gesteira, S. Longshaw, R. Canelas, R. Vacondio, A. Barreiro, and O. García-Feal, *DualSPHysics: Open-source parallel CFD solver based on Smoothed Particle Hydrodynamics (SPH),* Computer Physics Communications **187**, 204 (2015).

[82] J. J. Monaghan, *Smoothed particle hydrodynamics,* Reports on Progress in Physics **68**, 1703 (2005), 0507472v1 [astro-ph] .

[83] G. Jacobs and J. Hesthaven, *High-order nodal discontinuous Galerkin particle-in-cell method on unstructured grids,* Journal of Computational Physics **214**, 96 (2006).

[84] H. Moon, F. L. Teixeira, and Y. A. Omelchenko, *Exact charge-conserving scatter–gather algorithm for particle-in-cell simulations on unstructured grids: A geometric perspective,* Computer Physics Communications **194**, 43 (2015).

[85] I. Federico, S. Marrone, A. Colagrossi, F. Aristodemo, and M. Antuono, *Simulating 2D open-channel flows through an SPH model,* European Journal of Mechanics - B/Fluids **34**, 35 (2012).

[86] K. Olgaard, *Automated computational modelling for complicated partial differential equations*, Ph.D. thesis, Delft University of Technology (2013).

[87] M. Alnaes, A. Logg, K.-A. Mardal, O. Skavhaug, and H. Langtangen, *Unified framework for finite element assembly,* International Journal of Computational Science and Engineering (2009).

List of Figures

List of Tables