Desarrollo de un modelo avanzado de cálculo acoplado para el análisis del comportamiento en la mar de aerogeneradores flotantes

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

Jonathan Colom Cobb

Supervisors: Dr. Julio García-Espinosa
Dr. Borja Serván-Camas

Tutor: Dr. Xavier Martínez García

Departament de Ciència i Enginyeria Nàutiques
Universitat Politècnica de Barcelona

This dissertation is submitted for the degree of
Doctor of Philosophy in nautical engineering, marine and naval radioelectronics

Facultat de Nàutica June 2021

Development of an advanced model for seakeeping analysis of floating platforms

Jonathan Colom Cobb

Supervisors: Dr. Julio García-Espinosa
Dr. Borja Serván-Camas

Tutor: Dr. Xavier Martínez García

Departament de Ciència i Enginyeria Nàutiques
Universitat Politècnica de Barcelona

This dissertation is submitted for the degree of
Doctor of Philosophy in nautical engineering, marine and naval radioelectronics

Facultat de Nàutica June 2021

To my parents, Pep and Catherine, with love.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Jonathan Colom Cobb
July 2021

Acknowledgements

I owe great part of this thesis to all of the people who have been either helping me develop the thesis or given support throughout these past few years. This thesis would have been impossible without all of them.

First of all I would like to thank my supervisors, Julio García and Borja Serván. They gave me a great opportunity when I first came to Barcelona to work with them. Six years working side by side have been truly amazing, allowing me to learn different skills from two different but connected worlds: Programming skills and numerical analysis. The discussions at the blackboard in Borja's office are possibly the most fond memory I have working with them. It was at those moments where great ideas (and sometimes not that great too) were developed.

I cannot talk about blackboard discussions without also mentioning who got the whole marine group into the SL-PFEM method, Prashanth Nadukandi, or Augusto Maidana who was there throughout the whole thesis giving ideas while having coffee and even some biscuits! Jose Enrigue Gutiérrez has also been through quite a few of these discussions and always knows how to find interesting applications to the developed tools. Also, when we started developing the tool, we needed some sort of interface and here is were I want to thank the team at COMPASS I.S., in particular Rosa and Ovidi, they helped me with everything to do with the interface and even code-wise.

I would also like to thank my parents, Pep and Catherine, and my sister, Morgane, who have always tried to help as much as they can and given me unconditional support. My grandparents, Mimi and Chris, who would love to see me defend this thesis.
I am also thankful to my friends, I cannot mention all of you, but at times, without understanding much of what I do, you were all of great support. I do want to thank the two "perrenkes", Dani for helping me in some parts of the thesis and, together with Salva, for giving me such great moments these past years. Also, I have to thank everyone who passed through the NT3 building, but in particular I must thank Joel: Mate, we did it.

Six years side by side developing our thesis, but I must say that the best development we could achieve is our friendship.

Finally, I want to thank Ester for her love and being my biggest supporter in this phase of my life. Especially for being so patient with me, in particular this past year and a half while I have been writing this thesis. You cannot begin to imagine how grateful I am for giving me the strength I needed to finish this.

The research was developed thanks to the project XSHEAKS with reference ENE2014-59194-C2-1-R to which a predoctoral grant from MINECO was conceived.

Abstract

This thesis covers the implementation of a new tool to solve fluid problems capable of solving free-surface and fluid-solid interfaces. This tool is based on a Semi-Lagrangian Particle Finite Element Method (SL-PFEM) approach of solve the Navier-Stokes equations. In a similar fashion to the Particle Finite Element Method (PFEM) and its second generation version (PFEM-2), Lagrangian particles are used to solve the convection problem while a background mesh is used to solve the elliptic part of the Navier-Stokes equations. One of the objectives is that the tool must be efficient when calculating problems. This is managed by first developing the numerical scheme so that it is second order accurate in time and space and then optimising the implementation so that it can perform well in HPC environments.

To solve the general interface problems such as fluid-fluid, fluid-solid or the two combined, then an enrichment method is applied to the scheme. This, joined with an SL-PFEM variation of the Level-Set Method allows to accurately capture and compute these types of problems. This implementation leads to the ability to solve multi-body problems that can interact with multiple fluids.

Resumen

Esta tesis trata sobre la implementación de una nueva herramienta para resolver problemas fluidodinámicos tratando, en particular, los problemas de interfaces fluido-fluido y sólido-fluido. Esta herramienta está basada en el "Semi-Lagrangian Particle Finite Element Method" (SL-PFEM) para resolver las ecuaciones de Navier-Stokes. De manera similar al "Particle Finite Element Method" (PFEM) y su segunda versión (PFEM-2), partículas Lagrangianas son usadas para resolver el problema de convección mientras que una malla fija se usa para resolver la parte elíptica de las ecuaciones de Navier-Stokes. Uno de los objetivos es que la herramienta debe ser capaz de resolver los problemas de manera eficiente. Para ello, primero se desarrolla el esquema numérico para que sea un esquema de segundo orden tanto en espacio y tiempo y, luego, se optimiza la implementación para obtener un buen rendimiento en entornos de supercomputación.

Para resolver problemas de interfaces como los de fluido-fluido, solido-fluido o la combinación de ambas, un método de enriquecimiento es aplicado al esquema numérico. Esto, junto con una variación del método "Level-Set" aplicado al SL-PFEM permite capturar y calcular de forma precisa estos tipos de problemas. Además, esta implementación lleva a la capacidad de resolver problemas con más de un cuerpo que puedan interactuar con varios fluidos.

Table of contents

List of figures

List of tables

Nomenclature

Auxiliar Variables

$\mathbf{u}$	Convection velocity of the general Convection Equation
ν	Diffusion coefficient of the general Convection Equation
d	Distance between Node and Particle for the kernel type projectors
h	Element Size used for finite differences
q	Nodal characteristic lenth for the kernel type projectors
s	Source Term of the general Convection Equation

Eulerian Variables

$\mathbf{a}$	Acceleration at the eulerian frame
$\bar{\mathbf{D}}$	Divergence Matrix
$\bar{\mathbf{G}}$	Gradient Matrix
$\mathbf{x}$	Coordinates at the eulerian frame
$\bar{\mathbf{M}}$	Mass Matrix
$\bar{\mathbf{K}}$	Stiffness Matrix
P	Pressure at the eulerian frame
$\mathbf{u}$	Velocity at the eulerian frame

General Nomenclature

$\bar{\boldsymbol{\tau}}$	Stress tensor of the fluid
ρ	Density of a fluid
Γ_D	Contour of the domain with velocity dirichlet boundary conditions
Γ_N	Contour of the domain with Neumann boundary conditions
Γ_S	Contour of the domain with velocity slip boundary conditions
g	Gravity
$\Im\left(\right)$	Interpolation Operator used to map data from mesh to particles
$\mathbf{n}$	Normal vector of the domain's contour
Ω	Fluid domain
$\wp^{n+1}_{\{\lambda\}}\left\{\right\}$	Projection operator
ν	Kinematic viscosity of a fluid
Pe	Péclet Number
S	Strouhal number
t	Time variable

Lagrangian Variables

λ	Particle's label
$\mathbf{A}_\lambda$	Particle's acceleration at the Lagrangian frame
$\mathbf{X}_\lambda$	Particle's coordinates at the Lagrangian frame
$\mathbf{U}_\lambda$	Particle's velocity at the Lagrangian frame

Chapter 1

Introduction

1.1 Seakeeping hydrodynamics: State of the art and literature review

For the past decades, the basic method to determine the seakeeping abilities of a ship or floating structure is still based on the application of potential flow models that use the stokes perturbation theory (radiation/diffraction equations). Most of these models are solved in the frequency domain using the boundary element method [New92; New85; HS67; WAM] and are restricted to small body movements and small wave amplitudes. Recent models such as [SG13a] and [Ser16] present a different solution using a time domain scheme allowing to incorporate non-linear phenomena. The main problem of using the potential flow algorithms are the limitations they have. On one hand, the unresolved underlying physics are responsible, for example, for adding a dampening effect. On the other hand, these methods do not take into account the changes that are produced at the boundaries due to the body kinematics and changes in the free-surface shape.

In recent years, various authors have presented solutions where they apply Navier-Sotkes solvers to solve seakeeping problems [Cas+11], [Sim+13], [Car+07], [MCS10]. Even though these results are very promising, the practical application of these methods is very limited due to the high computational costs and, specially, due to the amount of time needed to obtain a solution.

Therefore, a need for efficient computational algorithms to solve high fidelity seakeeping hydrodynamics with fully non-linear free surface and large body movements is still a need.

The main challenge of all of these problems is how the free surfae (fluid-fluid interface) and body boundaries (solid-fluid interfaces) are tracked to impose the corresponding boundary conditions. In the case of two fluids, then two different flows are obtained that interact with each other via the interface. The same appears if the interface is formed due to a solid interacting with a fluid or, even more challenging, with another interface.

1.2 Fluid Interfaces: Known solver methodologies

In this thesis, fluid interfaces will refer either to a fluid-fluid interface (free surface interface) or a solid-fluid interface (body interface). Fluid interfaces are part of the fluid boundaries, and as such, the corresponding boundary conditions must be applied. To solve the full seakeeping hydrodynamics with the navier-stokes equations, one of the main difficulties is in fact the imposition of the boundary conditions at moving fluid interfaces. Depending on the method used, different schemes must be implemented. Two main variants of solving techniques can be found: Interface capturing and interface tracking methods.

1.2.1 Interface Capturing Methods

Interface capturing methods such as the level set [Bih+16; SS03; Set01; OS88; OF01; Cha+96] and the volume of fluid [HN81] use a fix mesh. The location of the interface is obtained by solving a new system and the interface is considered as a zone where a large change in the material properties appears. Interface capturing methods are commonly used when using the finite elements and finite volumes methods.

An initial way of solving these types of problems is by using the Finite Element Method (FEM) [ZT02; ZTN13]. This method discretises the domain of analysis into elements, therefore reducing the continuous problem to the number of unknowns on the nodes of the mesh. FEM though being a good tool to obtain continuous and smooth solutions it suffers to obtain discontinuous or rapidly changing solutions [Her09]. This difference of material properties translates into the discontinuity of the variables or of the gradients of the variables. These discontinuities are not properly captured when using the standard finite element method as its formulation does not allow for such problems. One way to tackle this problem is, as proposed at [Her09], by increasing the number of unknowns of the mesh elements that contain the interface of the two materials using enriched elements.

The Finite Volume Method is the most common way of solving fluid dynamics. The advantage over FEM is that it conserves and "balances" the numerical fluxes over the cell faces to obtain the finite volume equations [EGH00]. This way of formulating the problem makes it very attractive for fluid problems and has been widely used for fluid dynamics. For example, one of the most renown fluid dynamics solver, OpenFOAM [Ope; Wel+98], is based on this formulation. In order to solve interface problems when using FVM is the Volume of Fluid. This method, has the main advantage of conserving the mass but is incapable of accurately capturing the free surface as the cells only contain a percentage of each material, i.e. an interface is not defined within a cell only the amount of each material.

Level set methods are widely used, and have been developed and optimized for quite a long time. The standard LS method suffers from mass loss and numerical diffusion. The former comes from the inaccuracy to track the interface through a level set function, and the latter comes from the need of stabilisation of the pure convection equation. Improving these drawbacks require of sophisticated numerical techniques such as reinitialization of the LS function solving the Eikonal equation [Sus94] which in turn can result in expensive computations.

1.2.2 Interface Tracking Methods

Interface tracking methods are those that the computational domain moves with the shape of the interface and where the the fluid interface is a natural boundary of computational domain. Interface tracking methods are naturally used in a Lagrangian framework such as Smoothed Particle Hydrodynamics (SPH) [Mon94; LL10; GM82] and the Particle Finite Element Method (PFEM)[Ide+06a; Oña+11; IOP04; Ide+06b; Ide+08].

In a general manner the Lagrangian framework naturally captures interface problems accurately and tend to be a natural choice when trying to solve these types of problems. Lagrangian nodes or particles move with the fluid and, in particular, with the interface. This avoids the numerical diffusion introduced in the Eulerian framework by the stabilization of the convective terms but has to keep track of all the nodes/particles which can result to be computationally expensive. Keeping information of neighbouring nodes/particles or searching for these neighbours, can result to be quite expensive.

In the case of SPH methods, particles must constantly have the information of the neighbouring particles in order to define the interacting forces between each other. In fact, another problem appears at the domain boundaries, where solution strategies have to be implemented in order to properly capture when a particle has "collided" with the boundary.

Other problems appear when solving different fluid problems. When solving compressible flows, the particle pressure can be calculated from the local density using the equation of state. But, when solving an incompressible flow (such as water) then the equation of state of the fluid will lead to prohibitive time-steps that are extremely small [LL03; LL10]. Therefore, to overcome the problem with incompressible flows, the formulation can be changed in order to implement an artificial compressibility allowing to obtain a time derivative of the pressure [Mon94] and use more reasonable time-steps.

A final remark is the ability to use SPH for visual effects programs (VFX) such as Blender or Maya. It is done by assuming a decrease in accuracy when using simple kernels and state equations that allow large time-steps then solutions to fluid flows can be obtained in an extremely parallel way.

SPH methods, with other methods such as the Moving Particle Simulation [KTO95] and the Finite Point Method [Oña+96b; Oña+96a] belong to mesh-less methods. They all end up suffering from similar problems or limitations to the ones found in the SPH methods bringing up a need for an evolution of these methods. This came by using Finite Element types of meshes as a background. One of these methods was the Mesh-less Finite Element Method (MFEM) [IOD03; Ide+] which using a Delaunay tessellation [ICO03] obtains a set of shape functions without needing to construct a mesh. An evolution of these methods was proposed at [IOP04] with the Particle Finite Element Method (PFEM). This method includes the advantages of using lagrangian nodes in conjunction with a FE mesh. The idea is to discretise the problem in a way that the fluid domain can be viewed as material particles whose motion is tracked. Then, it computes the continuum equations on the FE mesh that is built from the lagrangian nodes.

PFEM over the years has proved to be successful for a variety of different problems, starting with the Navier-Stokes equations [AIO05] for Free-Surface flows [Lar+08] and fluid-structure interactions [Ide+08; Ide+06a; Pin+07]. In fact, particular applications have been performed such as industrial forming of materials [OFC14].

PFEM has many advantages but it also comes at a cost. It is necessary to reconstruct a mesh at every time-step as particles are constantly moving and can prove to be very costly. If not done, meshes could distort to a point where it becomes impossible to obtain a solution. This re-meshing problem brings up another problem which is a re-assembly problem. System matrices of the discrete problem must be re-built and re-assembled bringing the computational cost even higher.

From a parallelisation point of view, the re-meshing part brings up even more questions. Lagrangian nodes are moving from compute process to compute process meaning that element connectivities are constantly changing through the domain and forcing large amounts of communications between processes.

1.2.3 The semi-Lagrangian PFEM

After a brief review of the most commonly used methodologies used to compute fluid flows with interfaces, some conclusions can be extracted. Lagrangian methods naturally track the fluid interfaces avoiding to introduce numerical diffusion. On the other hand, Eulerain methods such as FEM and VOF are very suitable to solve numerically elliptic partial differential equations (such as Stoke´s equation) on a background mesh.

Semi-Lagrangian methods applied to solve the Navier Stokes equations try to take advantage of the Lagrangian and Eulerian frames, using the former to solve the convection, and the latter to solve a Stoke´s problem. In this sense, we can find different strategies. For instance, in [Gon01] the method of the charactersitics is combined with FEM; and in [Ide+12; Ide+13], Lagrangian particles are used in combination with FEM. In this thesis, an SL-PFEM method will be developed and tailored for seakeeping hydrodynamics.

This method is further developed in [Bec14; BIO15; Ide+14] where the basis of the method is presented and an initial error analysis [Ide+15; Gim+16] is performed and compared against the classic eulerian formulation. This method has proved to be more accurate in some of the cases and in particular, for interface problems. In fact, the big advantage of this method shown in the literature is the ability of obtaining accurate results using large time-steps and large mesh sizes.

In order to solve interface problems within the SL-PFEM, a semi-lagrangian approach of the Level Set method is used. This results in a combined method based on an interface capturing (level set) method with an interface tracking method based on lagrangian particles.

As with the initial PFEM, this method has been successfully applied to solve free surface problems [Bec14; BIO15; Ide+14; GGF19; GG15] and industrial applications such as mold filling or water-oil separation tank simulation [Gim+17]. Finally, a second order accurate scheme for PFEM-2 has been proposed at [Gim+19].

This method was viewed as a viable method to be applied in the marine and naval world when analysing the seakeeping behavior of floating objects [Nad+17]. It is from this last work where the current thesis starts off. It initially presents the X-IVAS (eXplicit

Integration along the Velocity and Acceleration Streamlines) scheme, which performs the particle's trajectory integration from the streamlines as it is a good approximation for strongly convected flows. It later goes on to show how this scheme, when solving wave problems is not appropriate. The fact is that the streamlines and the trajectories in a wave problem diverge rapidly. Therefore, a large reduction of the time-step is needed in order to obtain an accurate trajectory. [Nad+17] proposes an improvement of the X-IVAS scheme by using the acceleration field to better obtain the trajectories and correct the velocities.

1.3 Objectives

The aim of the thesis is the development of a fluid dynamics solver that is capable of solving interface problems, in particular fluid-fluid interfaces and fluid-solid interfaces, that will represent seakeeping hydrodynamics problems accurately. To reach this the final goal, a set of objectives are set that must be accomplished in a certain order.

The first objective is to obtain a solution of the Navier-Stokes equations using an SL-PFEM approach for single phase flows. This new integration scheme will have to overcome the problems stated at the previous section for wave problems and must be second order accurate.

The second objective is to further develop the ideas proposed for seakeeping hydrodynamic problems within the SL-PFEM framework and have a tool that can be capable of computing these types of problems. One of the main challenges of the seakeeping hydrodynamics problem is the imposition of the corresponding boundary conditions at the exact location of fluid interfaces. These can be considered as two interfaces: a fluid-fluid interface (free surface) and a solid-fluid interface (solid boundary). The developments obtained from this thesis will be used to simulate ship navigation in ice, where multiple bodies with different motions are found within the domain meaning that a new methodology to capture the solids-fluid interaction must be developed. Therefore, an important part is to be able to accurately capture the fluid-fluid and solid-fluid interfaces which, in turn, will allow to obtain good solutions to interface problems.

Finally, the proposed tool will be implemented in a parallel manner to maximize the compute resources of a day-to-day workstation and, in a larger scale, to make use of High Performance Computing (HPC) resources. This way, the final objective is to obtain an optimised tool that can be capable of solving large problems in an efficient manner.

To do this, a new framework for building this tool is created that is based on C++ and its latest standards and on MPI for parallel computing. This framework is a general framework on which other types of problems could also be implemented. The objective is that from a programming point of view, if someone wants to implement new schemes it must be easy to do and still maintain the parallel optimization. Optimally a user inserting a new scheme should not need to touch the lower level layer where the optimisations are found.

1.4 Publications

The work performed in this thesis has been used to publish one academic paper and to help the development of a book chapter.

- J. Colom-Cobb et al. "A second-order semi-Lagrangian particle finite element method for fluid flows". In: *Computational Particle Mechanics* 7.1 (2020), pp. 3–18
- J. García-Espinosa et al. "Development of New Lagrangian Computational Methods for Ice-Ship Interaction Problems: NICESHIP Project". In: *Computational Methods in Applied Sciences.* Vol. 54. Springer, 2020, pp. 121–153

1.5 Organisation

The structure of the thesis has been divided in six chapters and each chapter is dedicated to one or part of the proposed objectives of the thesis. Also each chapter contains two main parts: a first part dedicated to the theory of the problem in hand and a second part dedicated exclusively to verification and validation of the theory presented.

- **Chapter 2:** This chapter focuses on the development of the new scheme for single fluid problems. Here, the new scheme is presented and how second order accuracy in space and time is obtained. The fundamentals of the SL-PFEM with the new scheme are also presented and an error analysis is performed to prove the second order accuracy.
- **Chapter 3:** This chapter presents how interface problems will be tackled. It shows the methodology followed when trying to compute discontinuities in fluid problems. Once the methodology is presented it is then applied to free-surface problems, where the new methodology is implemented into the scheme proposed at chapter 2.

- **Chapter 4:** This chapter focuses on how the methodology to simulate interface problems presented at chapter 3 is used to solve fluid-solid interface problems. This chapter also presents the final combined problem where solid-fluid and free surface interfaces interact with each other.
- **Chapter 5:** The final chapter of the thesis will be focusing mainly on implementation details and how the new framework is structured. It presents some insights into how the optimised parallel algorithm is obtained.

Chapter 2

SL-PFEM for single phase flows

Along the thesis, Eulerian variables (mesh values) are written with lower case letters, particle variables are written with upper case letters and vectors are written with bold letters. Let $\mathbf{u}(x,t)$ be a velocity field, let $\{\lambda\}$ be a set of particles each of them identified with a label λ. The position occupied by each particle at time t is denoted by $\mathbf{X}_\lambda(t)$, and the velocity of the particle by $\mathbf{U}_\lambda(t)$. If the particle velocity is given by the velocity field $\mathbf{u}$, then $\mathbf{U}_\lambda(t) = \mathbf{u}(\mathbf{X}_\lambda(t), t)$, and the particles's trajectories are the pathlines defined by the velocity field $\mathbf{u}$.

2.1 The semilagrangian method

To solve the problem of fluid flows of a single phase the semi-lagrangian particle finite element method or SL-PFEM presented at the introduction is used. In this first subsection, the general methodology of the SL-PFEM is presented and it is discussed why this method is a good choice to solve the Navier-Stokes equations.

This will be done in different parts:

1. Instabilities due to the convection equation on the Eulerian frame.
2. Lagrangian formulation of the incompressible Navier-Stokes equations.
3. The SL-PFEM method for solving the incompressible Navier-Stokes equations.

The first two parts show the downfalls of using pure Eulerian or Lagrangian methods. The last part gives a brief introduction on how the SL-PFEM works.

2.1.1 Instabilities due to the convection equation on the Eulerian frame

When writing a steady convection-diffusion problem, we can present the equations as:

$$\mathbf{u} \cdot \nabla \, \phi - \nabla \cdot (\nu \, \nabla \, \phi) = s \tag{2.1}$$

where $\mathbf{u}$ is the convection velocity, ν the diffusion coefficient, s a source term and ϕ the unknown variable to solve.

The problem with the above equation appears when using numerical methods that discretise the computational domain into a mesh that has a finite number of elements and nodes. Methodologies found that use this type of discretisation are the ones that belong to the eulerian frameworks such as the Finite Difference Method, Finite Volume Method and the Finite Element Method. These methods, when discretising using standard second order numerical schemes for the convective term of equation 2.1, have problems with the convective term as it is only conditionally stable. To analyse the stability of these schemes, the Péclet number is defined. This number is defined as $Pe = \frac{\mathbf{u} \cdot h}{\nu}$, where h is the element size and relates the convective velocity with the diffusion of the problem. In fact, in [OM00] it shows that when $Pe > 1$ the scheme becomes unstable and produces spurious results and, in order to prevent this, stabilisation techniques are used. As a down-side of using these techniques, numerical diffusion is inserted into the results.

A good way of showing the inherent problem of having high péclet numbers is by choosing a classic 1D problem of the steady convection diffusion equations. This way, equation 2.1 is shown in 1D with the source term being $s = 0$:

$$u \, \phi_x - \nu \, \phi_{xx} = 0 \;\; in \; (0, L) \quad with \, a > 0 \tag{2.2}$$

We can integrate this problem using the standard second order shcemes for both the convection and diffusion terms such as the finite differences where h is a constant element size:

$$u \frac{\phi_{i+1} - \phi_{i-1}}{2 \, h} - \nu \frac{\phi_{i+1} - 2 \, \phi_i + \phi_{i-1}}{h^2} = 0 \tag{2.3}$$

And if we now implement and substitute into the equation the péclet number $Pe = \frac{u \cdot h}{\nu}$ then we obtain:

$$(Pe - 1) \, \phi_{i+1} + 2 \, \phi_i - (Pe + 1) \, \phi_{i-1} = 0 \tag{2.4}$$

Therefore allowing us to write this in a more meaningful and understandable way:

$$(\phi_{i+1} - \phi_i) = \frac{Pe + 1}{Pe - 1} (\phi_i - \phi_{i-1}) \tag{2.5}$$

What we can deduce is that if the Péclet number is $Pe < 1$ then the solution will be smooth but when $Pe > 1$ then we will start having numerical wiggling. An example that has an analytical solution is presented:

$$\begin{gathered} u\,\phi_x - \nu\,\phi_{xx} = 1 \;\; x \in [0,1] \\ \phi(0) = \phi(1) = 0 \\ Analytical\ solution: \\ \phi(x) = x - \frac{1}{u} \frac{\left(1 - e^{(u/\nu)x}\right)}{\left(1 - e^{u/\nu}\right)} \end{gathered} \tag{2.6}$$

The solutions of the example are shown in 2.1 where we can see that for high péclet numbers the aforementioned numerical wiggles appear. But if the problem to be solved has a predominant diffusive terms then the obtained numerical solution is in accordance with the analytical one.

In order to prevent the instabilities of the convection-diffusion equations, two paths can be taken: Reduce the mesh size or add stabilisation techniques. Now, both of these solutions do not fully fix the problem. By refining the mesh, the problem size becomes ever larger when velocities increase or when the diffusive part decreases; this makes it computationally inefficient. So, in order to prevent mesh refinement stabilisation techniques are used. In a broad manner, these techniques tend to add an artificial diffusion that ends up degrading the final result.

As a concluding remark, this thesis will develop an SL-PFEM capable of handling strong convective flows (high Péclet/Reynolds numbers) These problems need to be numerically accurate and computationally efficient therefore the implemented scheme will have to overcome some of the problems when using high Péclet numbers.

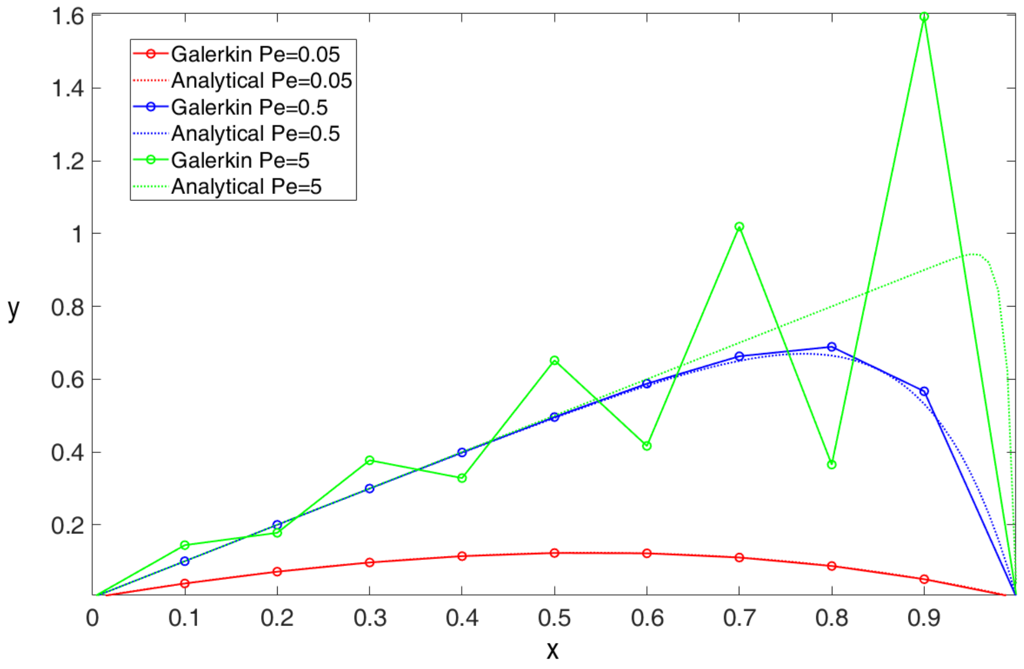


Fig. 2.1 Comparison of the 1D steady convection diffusion equation using different Péclet numbers

2.1.2 Lagrangian formulation of the incompressible Navier-Stokes equations

The incompressible Navier-Stokes equations can be solved by using a purely Lagrangian formulation. These types of formulations have the big advantage of solving the convective terms without having to use stabilisation techniques. In fact, the equations can be naturally presented as:

$$\begin{gathered} d_t\mathbf{u} = -\nabla \cdot \left(\frac{P}{\rho}\right) + \nu \cdot \Delta\mathbf{u} + \mathbf{f} \\ \nabla \cdot \mathbf{u} = 0 \end{gathered} \tag{2.7}$$

Where P is the fluid pressure, ρ is the fluid density, ν is the fluid kinematic viscosity, $\mathbf{f}$ are any external mass forces acting on the fluid and $\mathbf{u}$ is the fluid velocity. Equation 2.7 can be expressed as a velocity field depending on an acceleration field $d_t\mathbf{u} = \mathbf{a}$:

$$\mathbf{a} = -\nabla \cdot \left(\frac{P}{\rho}\right) + \nu \cdot \Delta\mathbf{u} + \mathbf{f} \tag{2.8}$$

Equation 2.7, shows how there are two main parts: At the left hand side, the material derivative which can be naturally solved on the Lagrangian framework and at the right hand side, the elliptic part of the equation, containing the pressure and viscous terms. Solving this second part of the Navier-Stokes equations in a Lagrangian framework by integrating the fluid particles dynamics is not as simple, since the acceleration field depends on the fluid velocity and the pressure. In order to ensure the divergence free condition of the flow field, the pressure must fulfil an integral equation on the domain. This proves to be very complicated when imposing the continuity condition in a Lagrangian framework. Instead, a slightly compressible flow approach is usually used, introducing an equation of state that relates the pressure with the density. Also, in order to compute the viscous term, space derivative must be evaluated, which is not trivial when using the Lagrangian frame.

2.1.3 Semi-Lagrangian integration of the incompressible Navier-Stokes equations

Up to now, the two different frameworks (Eulerian and Lagrangian) have shown that they are highly capable of solving different parts of the Navier-Stokes equations. Therefore, the combination of the two will be the fundamental basis of the SL-PFEM method. On one hand, the convective term will be solved over the lagrangian framework allowing to prevent the problem when using high Péclet numbers. On the other hand, the elliptic part of the equations (pressure and viscous terms) will be solved over the eulerian framework which is the type of problem that this framework is accurate and efficient at solving.

The equations to solve, presented at 2.7, must be completed by the appropriate boundary conditions that will allow to properly define the problem:

$$\mathbf{u} = \mathbf{u}^{Dirichlet} \qquad in \qquad \Gamma_D \times (0,t) \tag{2.9a}$$

$$\mathbf{u}\cdot\mathbf{n} = 0 \qquad \frac{\partial\left(\mathbf{u}\cdot\bar{\boldsymbol{\tau}}\cdot\mathbf{n}\right)}{\partial n} = 0 \qquad in \qquad \Gamma_S \times (0,t) \tag{2.9b}$$

$$\bar{\boldsymbol{\tau}}\cdot\mathbf{n} = -\left(\frac{P^{\cdot}}{\rho}\right)\mathbf{n} + \nu\frac{\partial\mathbf{u}}{\partial n} = 0 \qquad in \qquad \Gamma_N \times (0,t) \tag{2.9c}$$

$$\mathbf{u}\left(x,0\right) = \mathbf{u}_0\left(x\right) \qquad in \qquad \Omega \times (t=0) \tag{2.9d}$$

Γ_D is the part of the contour of the domain where the velocity field is imposed as a Dirichlet type of boundary condition. Γ_S is the part of the contour of the domain where the direction of the velocity is imposed and and normal derivatives of the tangential components of the velocity are null (also known as a slip condition). Γ_N is the contour part of the domain where the fluid stress is known (Neumann boundary condition) and, in fact, the boundary condition 2.9c is generally applied as an outlet of the fluid domain. $\mathbf{n}$ is the normal vector of the contour of the domain ($\partial\Omega$), t is the time variable and $\bar{\boldsymbol{\tau}}$ is the stress tensor of the fluid. The boundary condition 2.9c is generally applied as an outlet of the fluid domain.

So, with the boundary conditions defined, the SL-PFEM method can be applied in order to solve such equations. The main idea is to separate the equations into two parts thus, an operator split will be performed on the equations. Taking into account equation 2.8 where the acceleration field is defined by the pressure and viscous terms of the Navier-Stokes equations, then we can define $\{\lambda\}$ to be a set of fluid particles where the equation of motions can be presented as:

$$\begin{aligned} \frac{\mathbf{X}_\lambda(t)}{dt} &= \mathbf{U}_\lambda(t) \\ \frac{\mathbf{U}_\lambda(t)}{dt} &= \mathbf{A}_\lambda(t) \end{aligned} \tag{2.10}$$

Although a second order scheme will be presented later on and used along this thesis, for the sake of simplicity and just as an introduction to the SL-PFEM approach, the forward Euler scheme is used to integrate equation 2.10. As all the variables are known at time t^n then the new position of the fluid particle at time t^{n+1} can be easily calculated using any proposed scheme.

$$\begin{aligned} \frac{\mathbf{U}_\lambda^{n+1} - \mathbf{U}_\lambda^n}{\Delta t} &= \mathbf{A}_\lambda^{n+1} \\ \mathbf{X}_\lambda^{n+1} &= \mathbf{X}_\lambda^n + \Delta t \cdot \mathbf{U}_\lambda^n \end{aligned} \tag{2.11}$$

Where the particle's acceleration $\mathbf{A}_\lambda(t)$ is the rate of change in $\mathbf{U}_\lambda(t)$. With this, an acceleration field $\mathbf{a}(x,t)$ can be derived from the velocity field as:

$$\mathbf{a}(x,t) = d_t\mathbf{u}(x,t) \tag{2.12}$$

Being d_t the total derivative:

$$d_t\mathbf{u} = \partial_t\mathbf{u} + \mathbf{u}\nabla\mathbf{u} \tag{2.13}$$

Now, equation 2.11 can be split into two equations:

$$\begin{aligned} \frac{\mathbf{U}_\lambda^{n+1/2} - \mathbf{U}_\lambda^{n}}{\Delta t} &= 0 \\ \frac{\mathbf{U}_\lambda^{n+1} - \mathbf{U}_\lambda^{n+1/2}}{\Delta t} &= \mathbf{A}_\lambda^{n+1} = \mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right) \end{aligned} \tag{2.14}$$

In this case, the particle's acceleration field $\mathbf{A}_\lambda^{n+1}$ is in fact an interpolated variable from the background FEM mesh. To represent this, the interpolation is written as an evaluation of the mesh acceleration over the particle's position (equation 2.15).

$$\mathbf{A}_\lambda^{n+1} = \mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right) \tag{2.15}$$

Equations 2.14 show how an initial estimate of the velocity field on the lagrangian space is taken and then corrected using an updated acceleration field that needs to be computed to take into account the elliptic part of the Navier-Stokes equations. So, ideally the second equation of 2.14 on the lagrangian frame would have to be mapped on to the eulerian frame where the acceleration field could be solved and later used to correct the velocity field on the lagrangian frame. Therefore, with this consideration in mind, a projection step is defined which will pass through the variables on the lagrangian frame to the eulerian frame. This way, the projection operator is defined as:

$$\mathbf{u}\left(x,t\right) = \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda\left(t\right)\right\} \tag{2.16}$$

With this step defined, the second equation of 2.14 is presented on the eulerian frame using the projection step as:

$$\frac{\mathbf{u}^{n+1} - \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda^{n+1/2}\right\}}{\Delta t} = \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right)\right\} \tag{2.17}$$

Therefore, in order to solve the second equation of 2.14, the term $\mathbf{a}^{n+1}$ from 2.15 will be solved on the eulerian framework. This way, an equivalent eulerian equation is presented:

Where equation 2.17 is in fact the representation of equation 2.14-b mapped onto the background FEM mesh. This step is known as the projection step which uses the projection operator and will be in charge of passing through the variables on the lagrangian frame to the eulerian frame.

$$\mathbf{u}(x,t) = \wp_{\{\lambda\}}^{n+1}\{\mathbf{U}_\lambda(t)\} \tag{2.18}$$

The following condition is defined:

$$\phi^{n+1}(\mathbf{x}) = \wp_{\{\lambda\}}^{n+1}\left\{\phi\left(\mathbf{X}_\lambda^{n+1}\right)\right\} \tag{2.19}$$

Being ϕ an arbitrary variable defined on the eulerian framework. This condition will be presented with more detail but as a brief concept it is understood as a condition that applies to the projection and interpolation stages. This condition defines that the projector must be the inverse of the interpolator. In other words, that the projection of the nodal values interpolated at the particles returns the original values. Then, by complying with this condition, the equation shown at 2.17 can be presented as:

$$\frac{\mathbf{u}^{n+1} - \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda^{n+1/2}\right\}}{\Delta t} = \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right)\right\} \tag{2.20}$$

In other words, $\mathbf{u}^{n+1}(x) = \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda^{n+1}\right\}$. There is another outcome from the fulfilment of this condition. Figure 2.2 shows the conceptual implementation of a generic SL-PFEM. In general, two iterative loops are required to avoid any sort of splitting error. However, if the coherence condition is fulfilled, there is no need to iterate at the outer loop. This is because the correction of the particle's velocities, given by $\Delta t\mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right)$, will be projected back to the mesh as it is, with no further need for correction.

By having the variables mapped to the eulerian framework, it allows the acceleration field to be solved on this frame. To do this, a traditional iterative scheme based on the fractional step method is used using the Finite Element Method (FEM). The equations for this scheme will be presented further on as this is just a brief introduction to the method. Therefore, the acceleration field can be obtained as:

$$\mathbf{a}^{n+1} = \frac{\mathbf{u}^{n+1} - \mathbf{u}^{n+1/2}}{\Delta t} \tag{2.21}$$

And it can be interpolated to the lagrangian framework in order to perform the velocity correction.

$$\mathbf{U}_\lambda^{n+1} = \mathbf{U}_\lambda^{n+1/2} + \Delta t\mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right) \tag{2.22}$$

The full method is shown at figure 2.2 where the different steps are presented. As it can be seen the method is an implicit scheme when the outer iterative loop is performed.

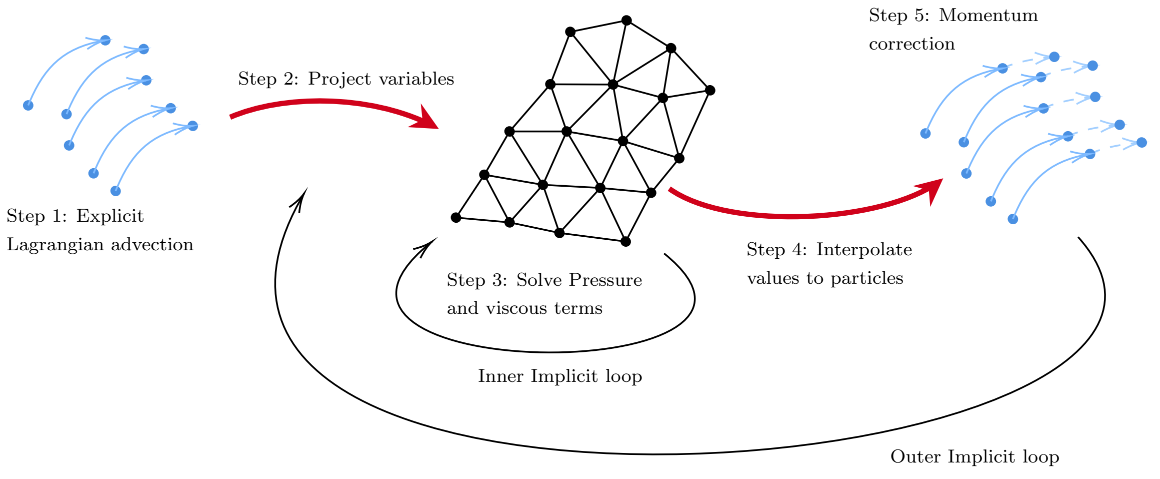

Step 1: Explicit Lagrangian advection
Step 2: Project variables
Step 3: Solve Pressure and viscous terms
Step 4: Interpolate values to particles
Step 5: Momentum correction
Inner Implicit loop
Outer Implicit loop

Fig. 2.2 Free surface interface (blue) over the background mesh (black)

To conclude this brief introduction to the method, figure 2.2 is recalled in order to present the different steps that the general SemiLagrangian Particle Finite Element Mehtod takes to solve the Navier Stokes equations (algorithm 1). In particular, when complying with the condition presented at 2.19 then the Outer Implicit Loop is no longer necessary as the splitting errors are not present.

The following sections will be presenting each of these steps and the schemes used to perform each of them. The last section of the chapter is an analysis of the scheme to prove that the final proposed second order scheme is truly second order accurate in space and time.

Algorithm 1: General algorithm used for the SL-PFEM

```
while (Time Loop: 1 to $n\,\Delta t$)
 {
   Start Time Step
   while (Outer Implicit Loop)
    {
      Lagrangian Convection
      Project from particles to FEM mesh
      while (Inner Implicit Loop)
       {
         Solve Pressure and Viscous terms using a FEM solver
       }
      Interpolate acceleration field from FEM mesh on Lagrangian particles
      Correct the particle velocity
    }
 }
```

2.2 Lagrangian convection of the particles

In the first section of this chapter, the general methodology of the SL-PFEM was presented. Now, in this section the lagrangian convection step from 2.2 is further developed to show the different available schemes. In particular, special attention is put into the second order convergence as one of the goals of the thesis is to obtain a second order accurate SL-PFEM.

The instabilities that arise from computing the convection part of the problem on the eulerian frame and the advantages of the lagrangian framework when solving the convective problem show the need to separate the convective part of the Navier Stokes equations. Equations 2.11 through 2.22 show how the Navier-Stokes equations are separated into the convective term on the lagrangian frame and the pressure and viscous terms on the eulerian frame.

Therefore, when using the lagrangian framework to solve the pure convection equation, the following equations of motion are to be solved:

$$\begin{aligned}\mathbf{X}_\lambda^{n+1} &= \mathbf{X}_\lambda^n + \int_{t^n}^{t^{n+1}} \mathbf{U}_\lambda(t) \\ \mathbf{U}_\lambda^{n+1} &= \mathbf{U}_\lambda^n + \int_{t^n}^{t^{n+1}} \mathbf{A}_\lambda(t)\end{aligned} \tag{2.23}$$

The approaches shown in this section compute the particle position using an explicit scheme and then compute the particle velocity using an implicit scheme.

2.2.1 Second Order Approach

In order to improve the scheme, a second order approach is implemented. The main idea is to implement a scheme that will compute the particle's pathline using an explicit scheme. Finally, a scheme that is suitable is the one found in molecular dynamics: the Velocity Verlet integrator ([Ver67],[Swo+82], [HLW03]).

$$\begin{aligned}\mathbf{X}_\lambda^{n+1} &= \mathbf{X}_\lambda^n + \Delta t \cdot \mathbf{V}_\lambda^n + \frac{\Delta t^2}{2}\mathbf{A}_\lambda^n \\ \frac{\mathbf{U}_\lambda^{n+1} - \mathbf{U}_\lambda^n}{\Delta t} &= \frac{1}{2}\left(\mathbf{A}_\lambda^n + \mathbf{A}_\lambda^{n+1}\right)\end{aligned} \tag{2.24}$$

Where, in this case $\mathbf{V}_\lambda^n$, $\mathbf{A}_\lambda^n$ and $\mathbf{A}_\lambda^{n+1}$ are the interpolated mesh velocity and accelerations over the particle's position:

$$\begin{aligned} \mathbf{V}_\lambda^n &= \mathbf{u}^n \left(\mathbf{X}_\lambda^n\right) \\ \mathbf{A}_\lambda^n &= \mathbf{a}^n \left(\mathbf{X}_\lambda^n\right) \end{aligned} \tag{2.25}$$

With this set of equations, we can now proceed to perform the operator split that allows to separate the convective term from the elliptic term of the Navier-Stokes equations:

$$\begin{aligned} \frac{\mathbf{U}_\lambda^{n+1/2} - \mathbf{U}_\lambda^n}{\Delta t} &= \frac{1}{2}\mathbf{a}^n \left(\mathbf{X}_\lambda^n\right) \\ \frac{\mathbf{U}_\lambda^{n+1} - \mathbf{U}_\lambda^{n+1/2}}{\Delta t} &= \frac{1}{2}\mathbf{a}^{n+1} \left(\mathbf{X}_\lambda^{n+1}\right) \end{aligned} \tag{2.26}$$

2.3 Particle to Mesh: Projectors

This section presents how data on the Lagrangian particles is mapped onto the FEM mesh of the Eulerian frame. If we recall figure 2.2 we can observe that the variables that are transported by the particles must be mapped onto the mesh in order to solve the Eulerian part of the Navier-Stokes equations presented at equation 2.8.

To do this, the particles and the mesh are communicated by simply having the particles contain information of the mesh such as the element they belong to. This allows to define a wide variety of operators that will map the information. For this thesis, the following projectors are defined:

- Kernel Based Projectors.
  - Shape Function Projection.
  - Radial based Projections: Shepard, Wendland, Exponential.
- Least Squares Projectors.
  - Local Node Least-Squares.
  - Local Element Least-Squares.
  - Global Least Squares based on Shape Function projection.

To finish off the section, the coherence condition is presented allowing us to define which projector is most adequate for the type of problem we are trying to solve.

2.3.1 Kernel Based Projectors

These types of projectors are based on using a weighting function to determine the nodal value of the variable transported by the particles. This function can be computed locally for each particle making it extremely parallel. The idea behind these projector types is to solve the following equation:

$$\phi_j = \frac{\sum\limits_{\lambda} f(x) \cdot \phi_\lambda}{\sum\limits_{\lambda} f(x)} \tag{2.27}$$

Where $f(x)$ is the weighting function, ϕ_j is the projected nodal value and ϕ_λ is the particle transported variable. The first type, are the ones that use the FEM shape functions as the weighting function: $f(x) = \mathbf{N}_\alpha(x)$.

Other types, are radial based projectors where the weighting functions depends on the distance between the node and the particle. One type of functions are those presented at [SS68], where one function is the inverse distance function: $f(x) = d^{-2}$ where d is the distance between node and particle. Another type of function that depends on the distance is one used in SPH [Lan+11]: the wendland kernel [Wen95]. This function that has been implemented is the 5th degree class two Wendland kernel, the same as in [Lan+11].

The main draw-back with these types of projections is that when the particle is exactly on the node, then the value of this particle is passed onto the node. This can be problematic, for instance, when the particles project a step function where a particle that falls exactly on a node with a value that is not the expected projected result will project its incorrect value to the node. Therefore, to overcome this problem, a gaussian type of projection is implemented:

$$f(x) = \mathbf{e}^{\frac{d^2}{0.16 \cdot q}} \tag{2.28}$$

Where q is defined as the nodal characteristic length.

2.3.2 Least Squares Projectors

For these types of projectors the idea is to minimise the error of the projection using the well known least-squares method. To do this, two different paths can be taken:

- Local Least Squares.
- Global Least Squares.

The first type of these projectors minimizes the error between the node value and a polynomial function of a certain degree. This is done locally for each node.

$$E = \sum_{\lambda} \left(\phi_j - P_k(\mathbf{X}_\lambda)\right)^2 \tag{2.29}$$

For example, a first order polynomial would be $\chi_1 + \chi_2\, x + \chi_3\, y$ for a two dimensional problem.

This same projector can be solved at a local element level. The idea is to obtain an approximating polynomial using the least-squares method for the local element and then obtain the nodal value from the calculated polynomial.

One of the problems with these types of projectors is that they are the least efficient ones, as they solves node-local or element-local systems that depend on the degree of the approximating polynomial function. In a first order approach and a 2D problem three degrees of freedom per node are to be solved but, when using a second order approximating polynomial, then the amount of degrees of freedom goes up to 6 per node.

This problem brings up a second one: to solve the system obtained using least squares a certain amount of particles are needed. For the 2D node-local projector using first order polynomials, 3 particles are needed per node independently if they are found in one element or another. This problem is even more severe when looking at the 2D element-local projector using first order polynomials where at least 3 particles are needed per element.

Remark: Element-local least squares projector is the first projector proposed that complies with the coherence condition which is demonstrated later on.

Finally, the second type of least-square projection, is based on the same idea, but this time the polynomial comes straight from the FEM interpolation by simply using the FE shape functions ($\mathbf{N}_j$). So, in order to perform the projection, the following error must be minimised:

$$E = \sum_{\lambda} \left(\phi_\lambda - \sum_j \phi_j \ \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \right)^2 \tag{2.30}$$

The main advantage is that this can be computed in a global way by solving one big system. In fact, when unfolding the minimization, the system that appears is:

$$\frac{\partial E}{\partial \phi} = -2 \cdot \sum_{\lambda} \left[\mathbf{N}_i \left(\mathbf{X}_\lambda \right) \left(\phi_\lambda - \sum_j \phi_j \ \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \right) \right] \tag{2.31}$$

And minimizing this means $\frac{\partial E}{\partial \phi} = 0$ so the system to solve is:

$$\sum_{\lambda} \left[\mathbf{N}_i \left(\mathbf{X}_\lambda \right) \left(\sum_j \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \right) \right] \phi_j = \sum_{\lambda} \mathbf{N}_i \left(\mathbf{X}_\lambda \right) \phi_\lambda \tag{2.32}$$

Which put in a matricial way it can be written as $\mathbf{M}\cdot\boldsymbol{\phi} = \mathbf{f}$ with $M = \sum_\lambda \mathbf{N}_i(\mathbf{X}_\lambda) \sum_j \mathbf{N}_j(\mathbf{X}_\lambda)$. In fact, the matrix $\mathbf{M}$ can be seen as a discrite approximation of the FEM mass matrix becoming a discrete version of a FEM projection.

The main disadvantage of this projector is the fact that a system must be assembled and later solved, meaning that a higher computational cost is needed compared to the kernel based projectors. The main advantage is that ideally only one particle per node is needed. Therefore, as long as one element that is connected to the node has a particle, then the system for that particular node will give a good solution.

Remark: When using the global least-square projector, when diagonalizing the shape-function projector is obtained. Therefore it can be stated that the shape-function projector is the lumped global least square projector. Finally, the global least square projector is the second projector that complies with the coherence condition being demonstrated at the following subsection.

2.3.3 Coherence condition

The previous subsection presented different types of projector schemes where it is mentioned that in particular two of them comply with the coherence condition. In this subsection, the coherence condition is presented and how the element-local least-squares projector and the global least-squares projector comply with this condition. It is first presented at the beginning of the chapter where the main advantage is the ability to prevent performing the fully implicit iterations (see figure 2.2).

The coherence condition defines that the projector must be the inverse operator of the interpolation. This means that a variable on the background mesh, when interpolated to the lagrangian particles and projected back to the eulerian mesh, then the resulting value must be the same as the initial one:

$$\phi_j^{n+1}(\mathbf{x}) = \wp_{\{\lambda\}}^{n+1}\left\{\sum_j \mathbf{N}_j\left(\mathbf{X}_\lambda^{n+1}\right)\phi_j\right\} \tag{2.33}$$

Let’s first consider the element-local least-squares:

$$E = \sum_\lambda \left(\phi_\lambda - P_k(\mathbf{X}_\lambda)\right)^2 \tag{2.34}$$

This projector obtains local polynomials for each element and the interpolation operator always uses the standard FE shape functions.Therefore, considering that both the obtained local polynomials and the interpolation polynomials are both linear within the element then the projection operation can be considered as the inverse of the interpolation operator.

Now let's consider the global least-squares:

$$\sum_{\lambda} \left[\mathbf{N}_i \left(\mathbf{X}_\lambda \right) \left(\sum_j \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \right) \right] \phi_j = \sum_{\lambda} \mathbf{N}_i \left(\mathbf{X}_\lambda \right) \phi_\lambda \tag{2.35}$$

If the particle variable ϕ_λ is substituted by the interpolated variable from the mesh $\phi_\lambda = \sum_j \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \phi^\star$, then the equation changes to:

$$\sum_{\lambda} \left[\mathbf{N}_i \left(\mathbf{X}_\lambda \right) \left(\sum_j \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \right) \right] \phi_j = \sum_{\lambda} \left[\mathbf{N}_i \left(\mathbf{X}_\lambda \right) \left(\sum_j \mathbf{N}_j \left(\mathbf{X}_\lambda \right) \right) \right] \phi^\star \tag{2.36}$$

Meaning that $\phi^\star = \phi_j$ forcing it to comply with the coherence condition.

2.3.4 Mesh to Particles: Interpolator

As shown at figure 2.2, another step that maps data between particles and the mesh exists. This operation is the inverse operation of the projector operator: an interpolation between the data found on the mesh to the particles. Recalling the scheme presented at equations 2.11 through 2.22 where the the Navier-Stokes equations are divided into the convective and elliptic parts, a final step is to update the particles transported velocity with the obtained Pressure and Viscous terms from the mesh. Hence the need for the interpolation operator, which will map the values obtained from the mesh back to the particles.

This interpolation is performed using the Finite Element Shape Functions, as particles contain information of the element they belong to. Therefore, the interpolation operator is defined at equation 2.37.

$$\phi_\lambda = \Im \left(\phi_j \right) = \sum_j \mathbf{N} \left(\mathbf{x} \right) \cdot \phi_j \tag{2.37}$$

Where ϕ_λ is the interpolated particle value, ϕ_j is the nodal value to interpolate and $\mathbf{N}\left(\mathbf{x}\right)$ is the FEM shape function.

2.4 Finite Element Approximation

This section presents the FEM discretisation of the elliptic part of the Navier-Stokes equations that are to be solved on the eulerian frame. The equations solve an acceleration field that contains the viscous and pressure gradient terms. The method to solve it is based on [SLC01], a traditional iterative scheme the comes from the fractional step method.

2.4.1 Differential form

To kick off this section, let us recall equation 2.26 where the convective part of Navier-Stokes equations are presented using the velocity Verlet integrator on the lagrangian frame.

$$\begin{aligned}\frac{\mathbf{U}_\lambda^{n+1/2}-\mathbf{U}_\lambda^{n}}{\Delta t}&=\frac{1}{2}\mathbf{a}^{n}\left(\mathbf{X}_\lambda^{n}\right)\\ \frac{\mathbf{U}_\lambda^{n+1}-\mathbf{U}_\lambda^{n+1/2}}{\Delta t}&=\frac{1}{2}\mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right)\end{aligned} \tag{2.38}$$

This time, and in a similar fashion to 2.14 and 2.22, the second equation of 2.26 is projected onto to the eulerian frame, being able to present it as:

$$\frac{\mathbf{u}^{n+1}-\wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda^{n+1/2}\right\}}{\Delta t}=\frac{1}{2}\wp_{\{\lambda\}}^{n+1}\left\{\mathbf{a}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right)\right\} \tag{2.39}$$

Defining $\mathbf{u}^{n+1/2}=\wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda^{n+1/2}\right\}$ and assuming that the coherence condition is fulfilled, then equation 2.39 becomes:

$$\mathbf{u}^{n+1}=\mathbf{u}^{n+1/2}+\frac{\Delta t}{2}\mathbf{a}^{n+1} \tag{2.40}$$

Where **a** is the acceleration field to be solved:

$$\mathbf{a}^{n+1}=-\nabla\left(\frac{P^{n+1}}{\rho}\right)+\nu\Delta\mathbf{u}^{n+1} \tag{2.41}$$

Substituting the acceleration field into 2.40, the equation to solve is presented as:

$$\frac{\mathbf{u}^{n+1}-\mathbf{u}^{n+1/2}}{\Delta t}=-\frac{1}{2}\nabla\left(\frac{P^{n+1}}{\rho}\right)+\nu\Delta\mathbf{u}^{\mathbf{n+1}} \tag{2.42}$$

And in a similar fashion to the fractional step scheme, the pressure equation is obtained by applying the divergence operator to equation 2.42. Taking into account the continuity condition $\nabla \cdot \mathbf{u}^{n+1} = 0$ then the obtained equation is:

$$\frac{\nabla \cdot \mathbf{u}^{n+1/2}}{\Delta t} = \frac{1}{2}\Delta\left(\frac{P^{n+1}}{\rho}\right) - \nabla \cdot \nu \Delta \mathbf{u}^{\mathbf{n+1}} \tag{2.43}$$

rewriting equation 2.43 allows to get the Pressure equation:

$$\Delta\left(\frac{P^{n+1}}{\rho}\right) = 2 \cdot \frac{\nabla \cdot \mathbf{u}^{n+1/2}}{\Delta t} - 2 \cdot \nabla \cdot \nu \Delta \mathbf{u}^{\mathbf{n+1}} \tag{2.44}$$

2.4.2 Spatial Discretisation

To obtain the discrete form we first must obtain a variational form of the problem proposed at 2.42 and 2.44 by performing the following steps:

1. Multiply the differential equations 2.42 and 2.44 by a function υ that is defined in a suitable space V for the momentum equation an by a function q defined also in a suitable space Q and that is null at the boundaries of the domain.
2. Integrate the equations 2.42 and 2.44 over the whole domain Ω.
3. Apply the Green equality in order to integrate by parts the laplatian operator.

Before performing this, it is worth to take a moment to define the two spaces of functions used V and Q. First, the notation is presented where $L^2(\Omega)$ is the space of integrable functions in a domain Ω, and $H_0^1(\Omega)$ denotes the Hilbert space of function vectors with a first derivative in $L^2(\Omega)$ that vanish at the contour of the domain Ω. Finally, a bold character is used to represent the vector counterpart of these spaces. Therefore, the two spaces used to define the functions are:

$$\begin{aligned} V &= H_0^1(\Omega) \\ Q &= \left\{ q \in L^2(\Omega) \middle| \int_\Omega q \, d\Omega = 0 \right\} \end{aligned} \tag{2.45}$$

So, with these definitions we are able to apply (1) and (2), obtaining the variational form of 2.42 and 2.44:

$$\int_\Omega \left[\mathbf{u}^{n+1}\right] \cdot \upsilon \; d\Omega = \int_\Omega \left[\mathbf{u}^{n+1/2} + \frac{\Delta t}{2}\left(-\nabla\left(\frac{P^{n+1}}{\rho}\right) + \nu\Delta\mathbf{u}^{n+1}\right)\right] \cdot \upsilon \; d\Omega \tag{2.46a}$$

$$\int_\Omega \left[\Delta\left(\frac{P^{n+1}}{\rho}\right)\right] \cdot q \; d\Omega = \left[\frac{2}{\Delta t}\int_\Omega \nabla \cdot \mathbf{u}^{n+1/2}\, d\Omega \cdot q \; - \nabla \cdot \nu\Delta\mathbf{u}^{n+1} \cdot q\right] \tag{2.46b}$$

Integrating by parts the second derivatives and the pressure gradients:

$$\begin{aligned}
\int_\Omega \nu\Delta\mathbf{u}^{n+1} \cdot \upsilon \; d\Omega &= \int_\Gamma \nu\frac{\partial\mathbf{u}^{n+1}}{\partial\mathbf{n}} \cdot \upsilon \; d\Gamma - \int_\Omega \nu\nabla\mathbf{u}^{n+1} \cdot \nabla\,\upsilon \; d\Omega \\
\int_\Omega \Delta\left(\frac{P^{n+1}}{\rho}\right) \cdot q \; d\Omega &= \int_\Gamma \frac{\partial\left(\frac{P^{n+1}}{\rho}\right)}{\partial\mathbf{n}} \cdot q \; d\Gamma - \int_\Omega \nabla\left(\frac{P^{n+1}}{\rho}\right) \cdot \nabla\, q \; d\Omega \\
\int_\Omega \nabla\left(\frac{P^{n+1}}{\rho}\right) \cdot \upsilon \; d\Omega &= \int_\Gamma \left(\frac{P^{n+1}}{\rho}\right) \upsilon \; \cdot \mathbf{n}\, d\Gamma - \int_\Omega \left(\frac{P^{n+1}}{\rho}\right) \nabla \cdot \, \upsilon \; d\Omega
\end{aligned} \tag{2.47}$$

And after integrating by parts, we get:

$$\int_\Omega \mathbf{u}^{n+1} \; \upsilon \; d\Omega = \left\{\begin{array}{l} \displaystyle\int_\Omega \mathbf{u}^{n+1/2} \; \upsilon \; d\Omega \\ \displaystyle + \int_\Omega \frac{\Delta t}{2}\left(\left(\frac{P^{n+1}}{\rho}\right)\nabla \cdot \, \upsilon \; - \nu\nabla\mathbf{u}^{n+1} \cdot \nabla\,\upsilon\right) d\Omega \\ \displaystyle + \int_\Gamma \left(\nu\frac{\partial\mathbf{u}^{n+1}}{\partial\mathbf{n}} \cdot \, \upsilon \; - \left(\frac{P^{n+1}}{\rho}\right) \, \upsilon\,\mathbf{n}\right) d\Gamma \end{array}\right\} \tag{2.48a}$$

$$\int_\Omega \nabla\left(\frac{P^{n+1}}{\rho}\right) \cdot \nabla\, q \; d\Omega = -\int_\Omega \left[\frac{2}{\Delta t}\nabla \cdot \mathbf{u}^{n+1/2}\, q\right] d\Omega + \int_\Gamma \frac{\partial\left(\frac{P^{n+1}}{\rho}\right)}{\partial\mathbf{n}} \cdot q \; d\Gamma \tag{2.48b}$$

With the two equations presented in their respective variational form, we can now approximate them by using the FEM. To do this, we will suppose that the domain Ω can be represented in a finite number of elements (in 2D triangles and in 3D tetrahedrons) that form an adequate mesh. In order to present the discrete version of the continuous variables, the following notation is used:

$$\begin{aligned} \mathbf{u}_h\,(x) &= \mathbf{u}_b\,\upsilon_{\,b}\,(x_j) \\ P_h\,(x) &= P_b\,q_{\,b}\,(x_j) \end{aligned} \tag{2.49}$$

Where $\mathbf{u}_h(x)$, $P_h(x)$ are the velocity and pressure discrete variables, and $\mathbf{u}_b$, P_b the values of the velocity and pressure variables approximated onto the nodes of the FE mesh. $\upsilon_b(x_j)$ and $q_b(x_j)$ are the FE shape functions that belong to the two spaces V and Q respectively. Therefore, first we can present the equations shown at 2.48 with the discrete variables, again using a set of shape functions υ_a and q_a which also belong to the respective spaces V and Q:

$$\int_\Omega \mathbf{u}_h^{n+1}\,\upsilon_a\,d\Omega = \left\{\begin{array}{l} \displaystyle\int_\Omega \mathbf{u_h}^{n+1/2}\,\upsilon_a\,d\Omega \\ \displaystyle+\int_\Omega \frac{\Delta t}{2}\left(\left(\frac{P_h^{n+1}}{\rho}\right)\nabla\cdot\upsilon_a - \nu\nabla\mathbf{u}_h^{n+1}\cdot\nabla\upsilon_a\right)d\Omega \\ \displaystyle+\int_\Gamma\left(\nu\frac{\partial\mathbf{u}_h^{n+1}}{\partial\mathbf{n}}\cdot\upsilon_a - \left(\frac{P_h^{n+1}}{\rho}\right)\upsilon_a\cdot\mathbf{n}\right)d\Gamma \end{array}\right\} \tag{2.50a}$$

$$\int_\Omega \nabla\left(\frac{P_h^{n+1}}{\rho}\right)\cdot\nabla q_a\,d\Omega = -\int_\Omega\left[\frac{2}{\Delta t}\nabla\cdot\mathbf{u}_h^{n+1/2}\,q_a\right]d\Omega + \int_\Gamma \frac{\partial\left(\frac{P_h^{n+1}}{\rho}\right)}{\partial\mathbf{n}}\cdot q_a\,d\Gamma \tag{2.50b}$$

And if we now insert the nodal variables, we get:

$$\int_\Omega \upsilon_b\,\upsilon_a\,\mathbf{u}_b^{n+1}\,d\Omega = \left\{\begin{array}{l} \displaystyle\int_\Omega \upsilon_b\,\upsilon_a\,\mathbf{u}_b^{n+1/2}\,d\Omega \\ \displaystyle+\int_\Omega \frac{\Delta t}{2}\left(\upsilon_b\nabla\cdot\upsilon_a\left(\frac{P_b^{n+1}}{\rho}\right) - \nu\nabla\upsilon_b\cdot\nabla\upsilon_a\,\mathbf{u}_b^{n+1}\right)d\Omega \\ \displaystyle+\int_\Gamma\left(\nu\frac{\partial\upsilon_b}{\partial\mathbf{n}}\cdot\upsilon_a\,\mathbf{u}_b^{n+1} - \upsilon_b\upsilon_a\cdot\mathbf{n}\left(\frac{P_b^{n+1}}{\rho}\right)\right)d\Gamma \end{array}\right\} \tag{2.51a}$$

$$\int_\Omega \nabla q_b\cdot\nabla q_a\left(\frac{P_b^{n+1}}{\rho}\right)d\Omega = \left\{\begin{array}{c} \displaystyle-\int_\Omega\left[\frac{2}{\Delta t}\nabla q_b\cdot q_a\right]\mathbf{u}_b^{n+1/2}\,d\Omega \\ \displaystyle+\int_\Gamma \frac{\partial q_b}{\partial\mathbf{n}}\cdot q_a\left(\frac{P_b^{n+1}}{\rho}\right)d\Gamma \end{array}\right\} \tag{2.51b}$$

In order to simplify these equations, a matricial notation will be inserted. The matrices are presented as:

$$\begin{aligned}\bar{\mathbf{M}}\,\boldsymbol{\phi} &= \int_\Omega \upsilon_b \upsilon_a \phi_b \, d\Omega \\ \bar{\mathbf{K}}\,\boldsymbol{\phi} &= \int_\Omega \nabla \upsilon_b \nabla \upsilon_a \phi_b \, d\Omega \\ \bar{\mathbf{G}}\,\boldsymbol{\phi} &= \int_\Omega \upsilon_b \nabla \upsilon_a \phi_b \, d\Omega \\ \bar{\mathbf{D}}\,\boldsymbol{\phi} &= \bar{\mathbf{G}}^T \phi_b\end{aligned} \tag{2.52}$$

But first a moment is needed to analyse how the contour parts of these equations are handled. First we have the contour part of the first equation of 2.51 which depends on the velocity and pressure variables:

$$\int_\Gamma \left(\nu \frac{\partial \upsilon_b}{\partial \mathbf{n}} \cdot \upsilon_a \mathbf{u}_b^{n+1} - \upsilon_b \upsilon_a \cdot \mathbf{n} \left(\frac{P_b^{n+1}}{\rho} \right) \right) d\Gamma \tag{2.53}$$

This part of the contour is 0 as defined at boundary conditions at the beginning of the chapter at equations 2.9.

Let's now look at the boundary integral given in 2.54. Problems can appear at the pressure equation where if simplifying the contour part (2.54) to 0 then the condition is naturally imposing zero pressure gradients at the contour. This can prove to be a problem with cases when the gravity is imposed where at the contour the pressure gradients are not 0.

$$\int_\Gamma \frac{\partial q_b}{\partial \mathbf{n}} \cdot q_a \left(\frac{P_b^{n+1}}{\rho} \right) d\Gamma \tag{2.54}$$

So in order to overcome this problem, first we assume that the contour component can be computed as:

$$\begin{aligned}\int_\Gamma \frac{\partial q_b}{\partial \mathbf{n}} \cdot q_a \left(\frac{P_b^{n+1}}{\rho} \right) d\Gamma &= \int_\Gamma \mathbf{n} \cdot \nabla q_b \cdot q_a \left(\frac{P_b^{n+1}}{\rho} \right) d\Gamma \\ &\approx \int_\Gamma \mathbf{n} \cdot q_b \cdot q_a \nabla \left(\frac{P_b^{n+1}}{\rho} \right) d\Gamma\end{aligned} \tag{2.55}$$

And, considering the integration by parts of the transposed of the gradient, written in a matricial manner:

$$\begin{gathered}\int_\Omega \nabla q_{\,b} \cdot q_{\,a} \mathbf{E}\, d\Omega = -\int_\Omega q_{\,b} \cdot \nabla q_{\,a} \mathbf{E}\, d\Omega + \int_\Gamma \mathbf{n} \cdot q_{\,b} \cdot q_{\,a} \mathbf{E}\, d\Gamma \\ \bar{\mathbf{G}}^T \mathbf{E} = -\bar{\mathbf{G}}\mathbf{E} + \bar{\mathbf{B}}\mathbf{E} \\ \bar{\mathbf{B}}\mathbf{E} = \left(\bar{\mathbf{D}} + \bar{\mathbf{G}}\right) \mathbf{E}\end{gathered} \tag{2.56}$$

Being:

$$\begin{gathered}\bar{\mathbf{M}}\mathbf{E} = \bar{\mathbf{G}} \left(\frac{P_b^{n+1}}{\rho}\right) \\ \bar{\mathbf{B}}\,\phi = \int_\Gamma \mathbf{n} \cdot q_{\,b} \cdot q_{\,a}\, \phi\, d\Gamma\end{gathered} \tag{2.57}$$

This way, taking into account how the contour terms are handled and the matricial notation, the equations from 2.51 can be written as:

$$\bar{\mathbf{M}}\,\mathbf{u}^{n+1} = \bar{\mathbf{M}}\,\mathbf{u}^{n+1/2} + \frac{\Delta t}{2}\,\bar{\mathbf{G}} \left(\frac{\mathbf{P^{n+1}}}{\rho}\right) - \frac{\Delta t}{2} \nu\, \bar{\mathbf{K}}\,\mathbf{u}^{n+1} \tag{2.58a}$$

$$\bar{\mathbf{K}} \left(\frac{\mathbf{P^{n+1}}}{\rho}\right) = -\frac{2}{\Delta t}\,\bar{\mathbf{D}}\,\mathbf{u}^{n+1/2} + \left(\bar{\mathbf{D}} + \bar{\mathbf{G}}\right)\,\bar{\mathbf{M}}^{\,-1}\,\bar{\mathbf{G}} \left(\frac{\mathbf{P^{n+1}}}{\rho}\right) \tag{2.58b}$$

As a final part, the equations presented at 2.58 are solved in a coupled manner by making them iterative:

$$\bar{\mathbf{M}}\,\mathbf{u}^{n+1,i+1} = \bar{\mathbf{M}}\,\mathbf{u}^{n+1/2} + \frac{\Delta t}{2}\,\bar{\mathbf{G}} \left(\frac{\mathbf{P^{n+1,i}}}{\rho}\right) - \frac{\Delta t}{2} \nu\, \bar{\mathbf{K}}\,\mathbf{u}^{n+1,i+1} \tag{2.59a}$$

$$\bar{\mathbf{K}} \left(\frac{\mathbf{P^{n+1,i+1}}}{\rho}\right) = -\frac{2}{\Delta t}\,\bar{\mathbf{D}}\,\mathbf{u}^{n+1/2} + \left(\bar{\mathbf{D}} + \bar{\mathbf{G}}\right)\,\bar{\mathbf{M}}^{\,-1}\,\bar{\mathbf{G}} \left(\frac{\mathbf{P^{n+1,i}}}{\rho}\right) \tag{2.59b}$$

Inserting 2.59a into 2.59b, the equations can be presented as:

$$\bar{\mathbf{M}}\,\mathbf{u}^{n+1,i+1} = \bar{\mathbf{M}}\,\mathbf{u}^{n+1/2} + \frac{\Delta t}{2}\,\bar{\mathbf{G}} \left(\frac{\mathbf{P^{n+1,i}}}{\rho}\right) - \frac{\Delta t}{2} \nu\, \bar{\mathbf{K}}\,\mathbf{u}^{n+1,i+1} \tag{2.60a}$$

$$\bar{\mathbf{K}} \left(\frac{\mathbf{P^{n+1,i+1}}}{\rho}\right) = -\frac{2}{\Delta t}\,\bar{\mathbf{D}}\,\mathbf{u}^{n+1,i+1} + \bar{\mathbf{G}}\,\bar{\mathbf{M}}^{\,-1}\,\bar{\mathbf{G}} \left(\frac{\mathbf{P^{n+1,i}}}{\rho}\right) \tag{2.60b}$$

Pressure stabilisation

The pressure equation, number 2 from equation 2.60, has an inherent stability problem. This problem has been analysed in a variety of publications such as ([CB00], [CB97], [SLC01]). These references explain how the pressure stability depends on Δt and that if a small enough time-step is selected then spurious effects can appear at the pressure variable. The stabilised formulation reads as:

$$(\tau + \Delta t)\,\bar{\mathbf{K}}\left(\frac{P^{n+1,i+1}}{\rho}\right) = \left\{\begin{aligned} -2\,\bar{\mathbf{D}}\,\hat{\mathbf{u}}^{n+1,i+1} + \tau\,\bar{\mathbf{G}}\,\bar{\mathbf{M}}_L^{-1}\,\bar{\mathbf{G}}\left(\frac{P^{n+1,i}}{\rho}\right) \\ +\Delta t\,\bar{\mathbf{K}}\left(\frac{P^{n+1,i}}{\rho}\right) \end{aligned}\right\} \tag{2.61}$$

This way, the pressure equation's stability does not depend on the time-step but on a variable τ that is defined by references to depend on the critical time-step. In a general manner, τ will be taken as the problem time-step but will be limited if the value goes below the critical time-step.

The derived pressure equation in this thesis is equivalent to 2.61 using a uniform $\tau = \Delta t$:

$$(2.60b)\ \bar{\mathbf{K}}\left(\left(\frac{\mathbf{P^{n+1}}}{\rho}\right)\right) = -\frac{2}{\Delta t}\bar{\mathbf{D}}\,\mathbf{u}^{n+1,i+1} + \bar{\mathbf{G}}\,\bar{\mathbf{M}}^{-1}\,\bar{\mathbf{G}}\left(\frac{\mathbf{P^{n+1,i}}}{\rho}\right) \tag{2.62}$$

By using $\tau = \Delta t$, the contour term of the pressure equation is retained allowing to recover the pressure gradients at the boundaries for hydrostatic dominated pressure problems. It has also been observed that the problems computed in this thesis have not suffered from small time-steps and thus have not needed to limit τ. In fact, the majority of the problems have the advantage of being able to use high time-steps without result deterioration allowing to state that τ will generally not need to be limited.

2.5 Error accumulation

SL-PFEM methods are prone to accumulate errors as information is passed from the two frameworks: Lagrangian particles onto the background Eulerian mesh and vice-versa. Those errors are originated in the projection and interpolation steps. And the accumulation of those errors could lead to several undesired effects such as the degradation of the rate of convergence of the numerical scheme, and the incapability to reach a steady state solution. This section aims at demonstrating that the rate of convergence of the proposed scheme is not degraded due to this phenomenon and that second order convergence and steady state solutions can be achieved.

To analyse the phenomenon, a few definitions are recalled: The velocity Verlet algorithm estimates the velocity and position at time $t^n = n\,\Delta t$ using a velocity field $\mathbf{u}\left(\mathbf{x}, t^n\right) = \mathbf{u}^n\left(\mathbf{x}\right)$, a corresponding acceleration filed $\mathbf{a}\left(\mathbf{x}, t^n\right) = \mathbf{a}^n\left(\mathbf{x}\right) = d_t\mathbf{u}^n\left(\mathbf{x}\right)$ and a set of particles with an initial condition $\mathbf{U}_\lambda{}^0 = \mathbf{u}^0\left(\mathbf{X}_\lambda{}^0\right)$, $\widehat{\mathbf{U}}_\lambda^0 = \mathbf{U}_\lambda^0$ and $\widehat{\mathbf{X}}_\lambda^0 = \mathbf{X}_\lambda^0$. As stated, this method achieves second order convergence:

$$\begin{aligned}\widehat{\mathbf{X}}_\lambda^{n+1} &= \widehat{\mathbf{X}}_\lambda^{n} + \Delta t\,\widehat{\mathbf{U}}_\lambda^{n} + \frac{\Delta t^2}{2}\mathbf{a}^n\left(\widehat{\mathbf{X}}_\lambda^{n}\right) = \mathbf{X}_\lambda^{n+1} + O\left(t^n\,\Delta t^2\right)\\ \widehat{\mathbf{U}}_\lambda^{n+1} &= \widehat{\mathbf{U}}_\lambda^{n} + \frac{\Delta t}{2}\left(\mathbf{a}^n\left(\widehat{\mathbf{X}}_\lambda^{n}\right) + \mathbf{a}^{n+1}\left(\widehat{\mathbf{X}}_\lambda^{n+1}\right)\right) = \mathbf{u}^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right) + O\left(t^n\,\Delta t^2\right)\end{aligned} \tag{2.63}$$

In order to evaluate the error accumulation over time, the error produced in the lagrangian integration of a given acceleration field will be estimated and so will the subsequent mesh projection and interpolation. This process is selected as the acceleration field to integrate will be the one obtained from the converged pressure and viscous terms calculated using 2.4 which is second order accurate. So, considering a space discretization using a Finite Element mesh, then the acceleration field on the lagrangian frame can be approximated using an interpolation field:

$$\mathbf{a}_\Im^n\left(\mathbf{X}_\lambda^n\right) = \sum_b \mathbf{N}^b\left(\mathbf{X}_\lambda^n\right)\mathbf{a}^n\left(x_b\right) = \sum_b \mathbf{N}^b\left(\mathbf{X}_\lambda^n\right)\mathbf{a}_b^n \tag{2.64}$$

The velocity on the mesh nodes is initiated from the initial particle velocity $\mathbf{U}_\lambda^0$ through the projection:

$$\left\{\widehat{\mathbf{u}}_b^0\right\} = \wp_{\{\lambda\}}^0\left\{\mathbf{U}_\lambda^0\right\} = \wp_{\{\lambda\}}^0\left\{\mathbf{u}^0\left(\mathbf{X}_\lambda^0\right)\right\} = \left\{\mathbf{u}_b^0 + \varepsilon^{P,0}\right\} \tag{2.65}$$

In this case, $\varepsilon^{P,0}$ is the projection error. The projected velocity is then interpolated back to the particles using the projected values:

$$\begin{aligned}\widehat{\mathbf{u}}^0\left(\mathbf{X}_\lambda^0\right) &= \Im\left(\widehat{\mathbf{u}}_b^0\right) = \sum_b \mathbf{N}^b\left(\mathbf{X}_\lambda^0\right)\widehat{\mathbf{u}}_b^0 = \sum_b \mathbf{N}^b\left(\mathbf{X}_\lambda^0\right)\left(\mathbf{u}_b^0 + \varepsilon^{P,0}\right) \\ &= \mathbf{u}_\Im^0\left(\mathbf{X}_\lambda^0\right) + \varepsilon_\Im^{P,0}\left(\mathbf{X}_\lambda^0\right)\end{aligned} \tag{2.66}$$

So, as it can be observed, the velocity and acceleration variables used in the verlet algorithm need to take into account the interpolation error $\varepsilon_\Im$:

$$\begin{aligned}\mathbf{u}_\Im^0\left(\mathbf{X}_\lambda^0\right) &= \mathbf{u}^0\left(\mathbf{X}_\lambda^0\right) + \varepsilon_\Im^{u,0}\left(\mathbf{X}_\lambda^0\right) \\ \mathbf{a}_\Im^0\left(\mathbf{X}_\lambda^0\right) &= \mathbf{a}^0\left(\mathbf{X}_\lambda^0\right) + \varepsilon_\Im^{a,0}\left(\mathbf{X}_\lambda^0\right)\end{aligned} \tag{2.67}$$

Then, the Verlet integrator can be presented taking into account the different errors for the first time-step:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^1 &= \mathbf{X}_\lambda^0 + \Delta t\mathbf{u}^0\left(\mathbf{X}_\lambda^0\right) + \Delta t\varepsilon^{u,0}\left(\mathbf{X}_\lambda^0\right) + \Delta t\varepsilon_\Im^{P,0}\left(\mathbf{X}_\lambda^0\right) \\ &\quad + \frac{\Delta t^2}{2}\mathbf{a}^0\left(\mathbf{X}_\lambda^0\right) + \frac{\Delta t^2}{2}\Delta t\varepsilon^{a,0}\left(\mathbf{X}_\lambda^0\right) \\ \tilde{\mathbf{U}}_\lambda^1 &= \mathbf{U}_\lambda^0 + \frac{\Delta t}{2}\left(\mathbf{a}^0\left(\mathbf{X}_\lambda^0\right) + \mathbf{a}^1\left(\tilde{\mathbf{X}}_\lambda^1\right)\right) + \frac{\Delta t}{2}\left(\varepsilon^{a,0}\left(\mathbf{X}_\lambda^0\right) + \varepsilon^{a,1}\left(\tilde{\mathbf{X}}_\lambda^1\right)\right)\end{aligned} \tag{2.68}$$

Taking into account the projector $\wp_{\{\lambda\}}^n\{x\}$ and the interpolator $\Im(x)$ operators are second order accurate, then equations from 2.68 can be presented substituting ε for the estimated error:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^1 &= \overbrace{\mathbf{X}_\lambda^0 + \Delta t\mathbf{u}^0\left(\mathbf{X}_\lambda^0\right) + \frac{\Delta t^2}{2}\mathbf{a}^0\left(\mathbf{X}_\lambda^0\right)}^{\widehat{\mathbf{X}}_\lambda^1 = \mathbf{X}_\lambda^1 + O\left(\Delta t^3\right)} + \overbrace{\Delta t\varepsilon^{u,0}\left(\mathbf{X}_\lambda^0\right)}^{O\left(\Delta t\, h_e^2\right)} + \overbrace{\Delta t\varepsilon_\Im^{P,0}\left(\mathbf{X}_\lambda^0\right)}^{O\left(\Delta t\, h_e^2\right)} \\ &\quad + \underbrace{\frac{\Delta t^2}{2}\Delta t\varepsilon^{a,0}\left(\mathbf{X}_\lambda^0\right)}_{O(\Delta t^2\, h_e^2)} \\ \tilde{\mathbf{U}}_\lambda^1 &= \overbrace{\mathbf{U}_\lambda^0 + \frac{\Delta t}{2}\left(\mathbf{a}^0\left(\mathbf{X}_\lambda^0\right) + \mathbf{a}^1\left(\widehat{\mathbf{X}}_\lambda^0\right)\right)}^{\widehat{\mathbf{U}}_\lambda^1 = \mathbf{U}_\lambda^1 + O\left(\Delta t^3\right)} + \overbrace{\frac{\Delta t}{2}\left(\mathbf{a}^1\left(\tilde{\mathbf{X}}_\lambda^1\right) - \mathbf{a}^1\left(\widehat{\mathbf{X}}_\lambda^1\right)\right)}^{\sim\frac{\Delta t}{2}\left(h_e^2\Delta t\right)\nabla\mathbf{a}^1} \\ &\quad + \underbrace{\frac{\Delta t}{2}\left(\varepsilon^{a,0}\left(\mathbf{X}_\lambda^0\right) + \varepsilon^{a,1}\left(\tilde{\mathbf{X}}_\lambda^1\right)\right)}_{O(\Delta t\, h_e^2)}\end{aligned} \tag{2.69}$$

Equations presented at 2.69 can be simplified:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^1 &= \widehat{\mathbf{X}}_\lambda^1 + \Delta t\, O\left(h_e^2\right) = \mathbf{X}_\lambda^1 + O\left(\Delta t^3\right) + O\left(\Delta t\, h_e^2\right)\\ \tilde{\mathbf{U}}_\lambda^1 &= \widehat{\mathbf{U}}_\lambda^1 + \Delta t\, O\left(h_e^2\right) = \mathbf{U}_\lambda^1 + O\left(\Delta t^3\right) + O\left(\Delta t\, h_e^2\right)\end{aligned} \tag{2.70}$$

As it can be observed, the error committed by the integration, projection and interpolation step are of order $O\left(\Delta t^3\right) + O\left(\Delta t\, h_e^2\right)$. In fact, the error can be presented as $\Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)$ meaning that, for the first time-step of sizeΔt, the error is second order accurate in time and space. Now, using an induction process, the accumulated error when performing $\{n+1\}$ steps will be presented. To do this, first the initial assumption for step n is shown:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^n &= \mathbf{X}_\lambda^n + n\,\Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\\ \tilde{\mathbf{U}}_\lambda^n &= \mathbf{U}_\lambda^n + n\,\Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\end{aligned} \tag{2.71}$$

Then, to obtain the error accumulation for $\{n+1\}$ steps the velocity Verlet integrator for this step can be presented as:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^{n+1} &= \tilde{\mathbf{X}}_\lambda^n + \Delta t\mathbf{u}^n\left(\tilde{\mathbf{X}}_\lambda^n\right) + \frac{\Delta t^2}{2}\mathbf{a}^n\left(\tilde{\mathbf{X}}_\lambda^n\right) + \Delta t\varepsilon^{u,n}\left(\mathbf{X}_\lambda^n\right)\\ &\quad + \Delta t\varepsilon_{\Im}^{P,n}\left(\mathbf{X}_\lambda^n\right) + \frac{\Delta t^2}{2}\Delta t\varepsilon^{a,n}\left(\mathbf{X}_\lambda^n\right)\\ \tilde{\mathbf{U}}_\lambda^n &= \tilde{\mathbf{U}}_\lambda^n + \frac{\Delta t}{2}\left(\mathbf{a}^n\left(\mathbf{X}_\lambda^n\right) + \mathbf{a}^{n+1}\left(\tilde{\mathbf{X}}_\lambda^n\right)\right) + \frac{\Delta t}{2}\left(\varepsilon^{a,n}\left(\mathbf{X}_\lambda^n\right) + \varepsilon^{a,n+1}\left(\tilde{\mathbf{X}}_\lambda^{n+1}\right)\right)\end{aligned} \tag{2.72}$$

If we now take into account the order estimation used in 2.69 and substitute $\tilde{\mathbf{X}}_\lambda^n$ and $\tilde{\mathbf{U}}_\lambda^n$ for the assumed error accumulation until step n at equation 2.71, then equation 2.72 can be presented with the error accumulation until step n:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^{n+1} &= \overbrace{\mathbf{X}_\lambda^n + \Delta t\mathbf{u}^n\left(\mathbf{X}_\lambda^n\right) + \frac{\Delta t^2}{2}\mathbf{a}^n\left(\mathbf{X}_\lambda^n\right)}^{\mathbf{X}_\lambda^{n+1}} + n\,\Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\\ &\quad + \Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\\ \tilde{\mathbf{U}}_\lambda^n &= \underbrace{\mathbf{U}_\lambda^n + \frac{\Delta t}{2}\left(\mathbf{a}^n\left(\mathbf{X}_\lambda^n\right) + \mathbf{a}^{n+1}\left(\tilde{\mathbf{X}}_\lambda^n\right)\right)}_{\mathbf{U}_\lambda^{n+1}} + n\,\Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\\ &\quad + \Delta t\left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\end{aligned} \tag{2.73}$$

Which, after substituting the values, returns:

$$\begin{aligned}\tilde{\mathbf{X}}_\lambda^{n+1} &= \mathbf{X}_\lambda^{n+1} + \{n+1\}\ \Delta t \left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\\ \tilde{\mathbf{U}}_\lambda^{n+1} &= \mathbf{U}_\lambda^{n+1} + \{n+1\}\ \Delta t \left(O\left(\Delta t^2\right) + O\left(h_e^2\right)\right)\end{aligned} \tag{2.74}$$

This shows the error accumulation over $\{n+1\}$ steps and proves that the proposed scheme, when using second order accurate projectors and interpolators, is second order accurate over space and time.

Higher order projectors and interpolators

When performing the projection step or the interpolation step one can be tempted to use a higher order projector or interpolator operator so that a higher order of convergence is obtained. This is not valid as long as a second order FEM is used to obtain the acceleration field.

The latter second order error will be interpolated and projected as such, regardless of the order of the projectors and interpolators used. Therefore using higher than second order projectors and interpolators, while possibly leading to some improvement, is not mandatory when using the velocity Verlet algorithm.

As a concluding remark, to obtain second order accuracy in time and space using the velocity Verlet integrator, the projector and interpolator must be second order accurate in space. If a higher order accuracy is desired, then all of the schemes must be elevated to this order: Integration of the equations of motion on the lagrangian frame, projection onto the mesh nodes, FEM integration of pressure and viscous terms, and the interpolation over particles.

2.6 Verification and Validation

2.6.1 Projectors: Verification of the Coherence condition

An initial verification is to check how the different projectors perform with different types of functions. In fact, the coherence condition can be observed by performing projection to mesh and interpolation to particles iterations. This is a good way of choosing the correct projector for the desired problem.

The function used has to be one that mimics the behaviour of the velocity field for single phase flows. Therefore, a sinusoidal wave seems to be a good option as it is a smooth and continuous function. The main idea is to observe how such a function behaves when performing projection and interpolation operations in a repetitive manner. To perform this, the different projectors presented are used in a 1D problem (See figure 2.3 for the sinusoidal wave.)

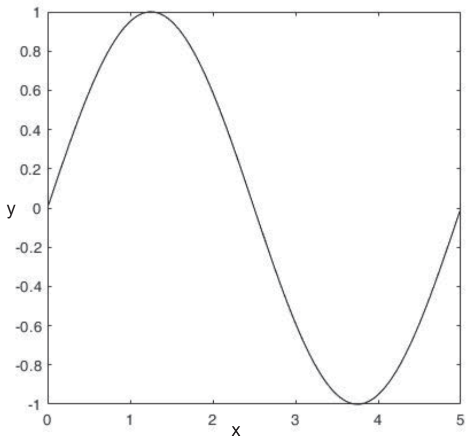


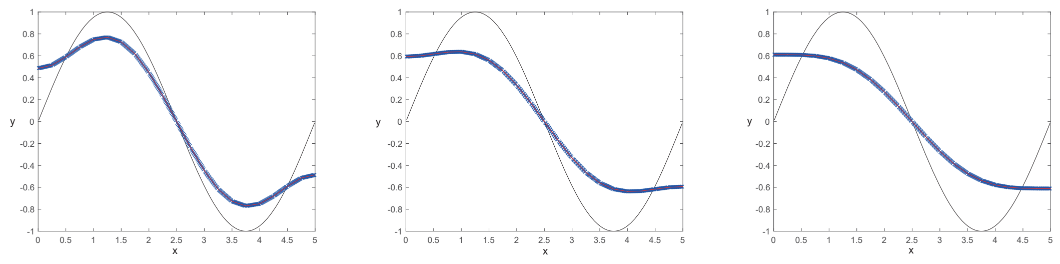
Fig. 2.3 Initial sinusioidal wave function for the verification.

The parameters used for this problem are a domain of size $[0, 5]$ which contains a sinusoidal function of amplitude 1 and is discretised using a total of 20 equal-sized elements (21 nodes). Each element contains a total of 20 particles.

In order to see the difference when complying or not with the coherence condition 50 iterations are performed and the errors are shown at figure 2.9. In this case, it is expected to see great performance from the global least squares and the two element local least squares as they are the only two to comply with the coherence condition.

So, a total of eight different projectors are tested: three kernel types (inverse distance, gaussian and shape function), two node local least squares (first and second order), two element local least squares (first and second order) and finally the global least squares.

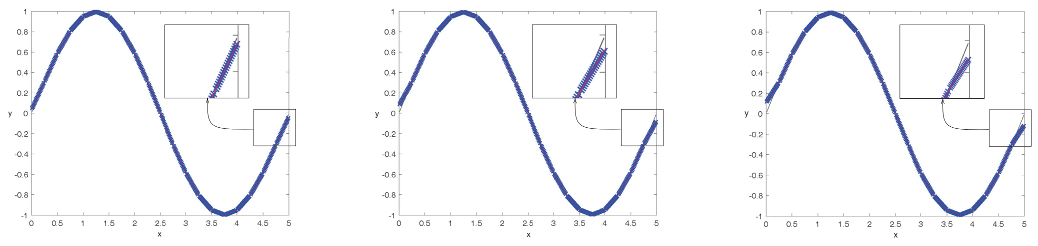
The first tested projectors are the kernel based ones shown at figure 2.4. As it can be observed, the final function (blue) has degraded enough to not even represent correctly the sinusoidal shape. Also, one problem starts to appear with the kernel projectors: the error produced at the boundaries. These projectors suffer when projecting values at the boundary. This produces an error that, added with the iterative projector-interpolator error, highly degrades the solution at the boundary.



(a) Sheppard inverse distance (b) Sheppard gaussian (c) Shape function

Fig. 2.4 Sinusoidal function projected using Kernel projectors

After observing this behaviour, tests to see if this appears at the initial iterations where conducted. Figure 2.5 shows the sinusoidal result after performing only one iteration and clearly showcases the problem in hand.



(a) Sheppard inverse distance (b) Sheppard gaussian (c) Shape function

Fig. 2.5 Sinusoidal function projected using Kernel projectors after one iteration

The last set of figures show the committed error using the least-squares methods. As expected, the global least squares (figure 2.6) and the element local least squares (figure 2.7) have no error accumulation apart from the error introduced in the first iteration.

Figure 2.9 clearly shows that these two projectors only commit the initial error at the first iteration and after that, the error does not increase.

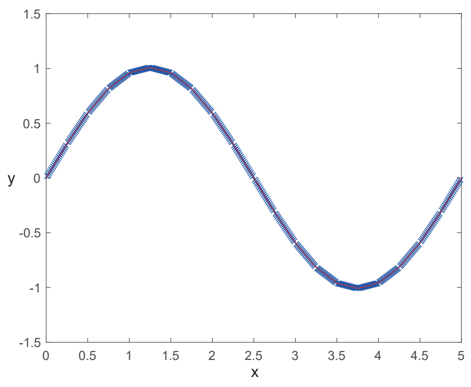
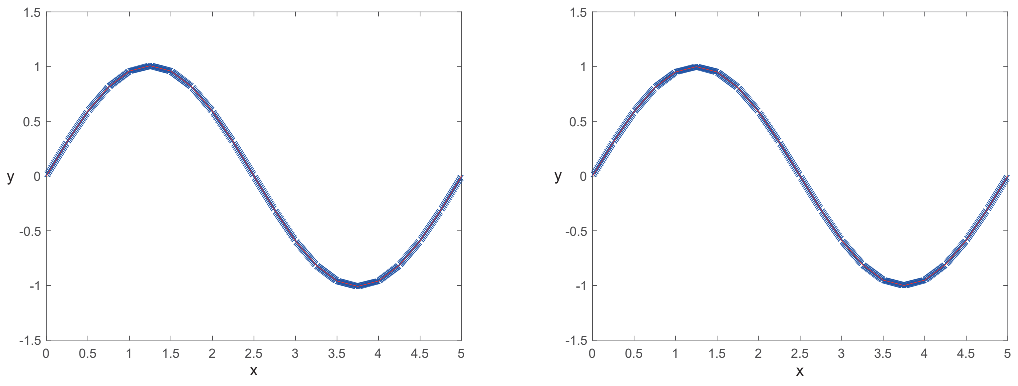


Fig. 2.6 Sinusoidal function projected using Global Least Squares



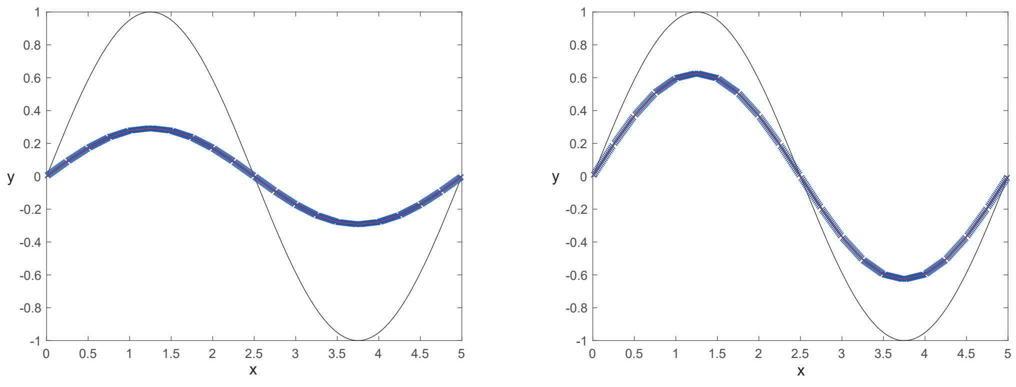
(a) First order

(b) Second order

Fig. 2.7 Sinusoidal function projected using element local least squares

Finally the node local least square is presented at figure 2.8. It can be seen that some error does appear, but compared to the kernel projectors, the sinusoidal shape is maintained even at the boundary of the problem.

Finally, figure 2.9 presents the Root Mean Square Error (RMSE) of the different projectors after performing 50 projection-interpolation iterations over the sinusoidal function. It shows that projectors complying with the coherence condition do not obtain errors over the iterations and, how the node local second order projector improves the committed error.



(a) First order (b) Second order

Fig. 2.8 Sinusoidal function projected using node local least squares

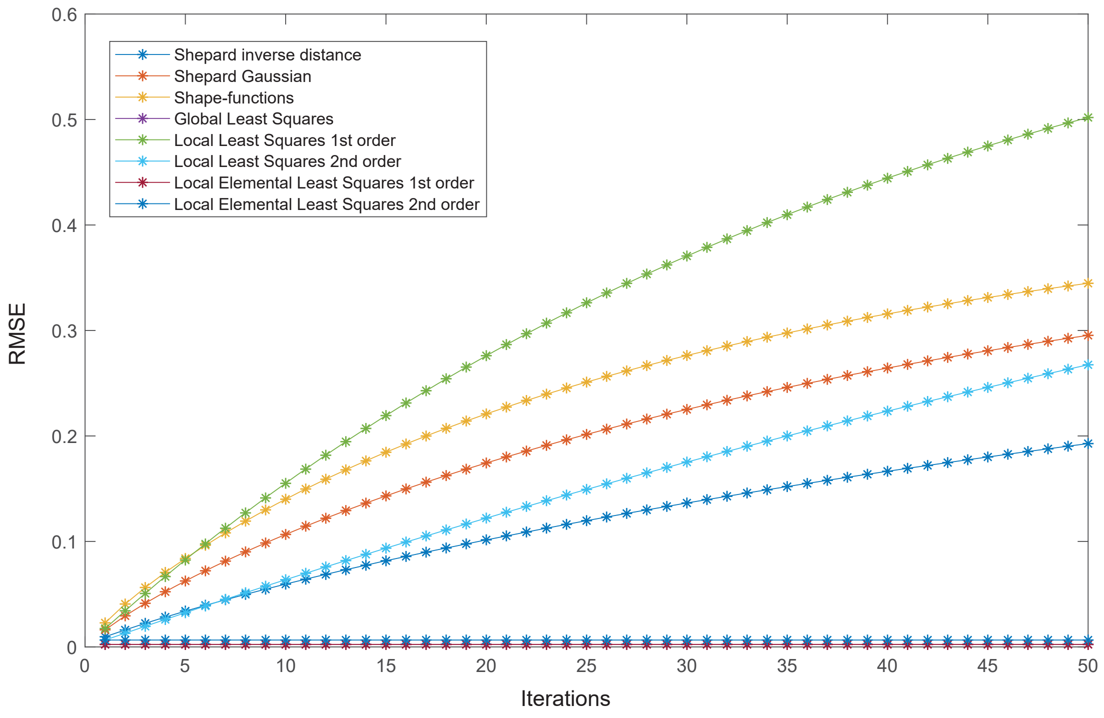


Fig. 2.9 RMSE of the sinusoidal wave after projecting and interpolating 50 times with the different projectors.

Conclusions

It is important to note the results obtained for a smooth function when being projected to the background FE mesh, especially when deciding which projector to use. The velocity field, which is within the scope of such functions, can be projected using any of the least-squares methods as they will approximate these types of problems correctly. But as this variable depends on the acceleration field then it is convenient that the projector used complies with the coherence condition in order to prevent the outer iterations of 2.2.

But problems with the velocity field can appear when using projectors that add numerical diffusion or numerical oscillations such as the Kernel type of projectors. This is worsened by using a projection-interpolation scheme that does not fulfil the coherence condition (ending with a larger number of necessary iterations) or if the problem requires a larger number of iterations for convergence purposes. By performing more projection stages that add diffusion or oscillations, the final result can be severely degraded.

2.6.2 Linear Wave trajectories

As it has been pointed out at the first chapter, for non-dominant convective flows the explicit streamlines computed at the beginning of the time step are not good approximates to pathlines and therefore small time steps are required to achieve accurate results [Nad+17]. The flow field induced by a linear wave is a good example used to demonstrate this point. And, if information regarding the acceleration is introduced, the required time step can be increased (see Figure 2.10).

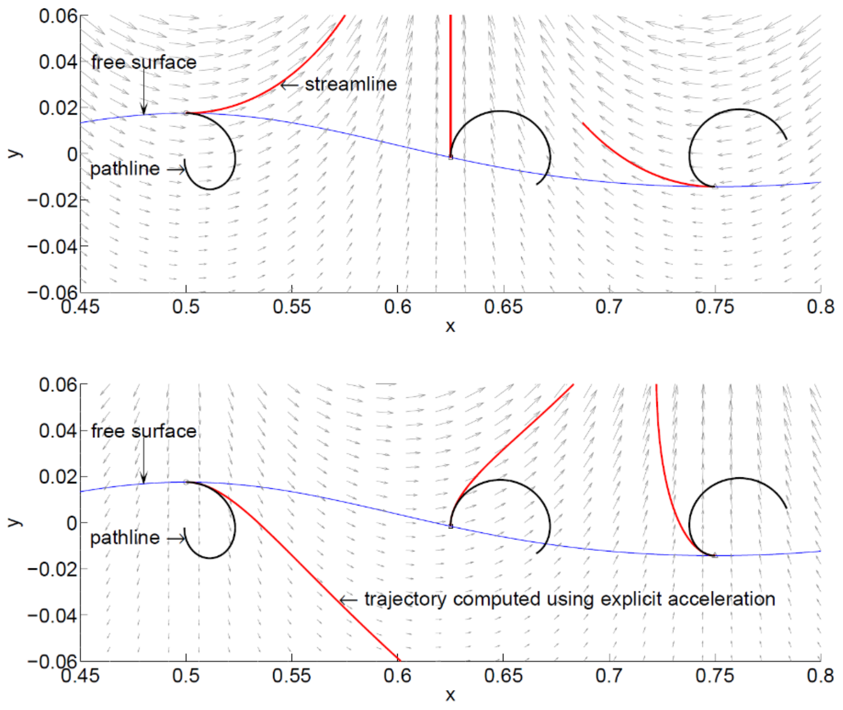


Fig. 2.10 Approximation of pathlines of fluid particles under an acceleration field induced by a linear wave. Top; using explicit streamlines. Down: computing trajectories with explicit acceleration. Figure extracted from [Nad+17].

In this section, the velocity Verlet algorithm is used to estimate the trajectory of a particle moving in the acceleration field induced by the velocities of a linear wave. The 2D velocity potential of an Airy wave with mean free surface located at $y = 0$ is given by:

$$\varphi(x, y, t) = \frac{A\,g}{\omega} \frac{\cosh(K\,(H + y))}{\cosh(K\,H)} \sin(K\,x - \omega\,t) \tag{2.75}$$

Where φ is the velocity potential, A is the wave amplitude, $K = 2\pi/L$ is the wave number and L is the wave length, H is the water depth, $\omega = 2\pi/T$ is the wave frequency and T is the wave period, x and y are the vertical and horizontal spatial coordinates, t is time.The velocity field is obtained as the gradient of the velocity potential $(u_x, u_y) = (\varphi_x, \varphi_y)$. Then, the corresponding acceleration field is obtained via $(a_x, a_y) = d_t (u_x, u_y)$. Figure 2.10 shows the velocity field induced by a linear wave. Using the velocity Verlet integrator we obtain the following equations for the position of the particle and its velocity components:

$$
\begin{aligned}
\mathbf{X}_\lambda^{n+1} &= \left\{ \begin{array}{l} \mathbf{X}_\lambda^n + \Delta t \varphi_x \left(\mathbf{X}_\lambda^n, t^n\right) \\ + \dfrac{\Delta t^2}{2} \left(\begin{array}{r} \varphi_{xt} \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_x \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_{xx} \left(\mathbf{X}_\lambda^n, t^n\right) \\ + \varphi_y \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_{xy} \left(\mathbf{X}_\lambda^n, t^n\right) \end{array} \right) \end{array} \right\} \\
\mathbf{Y}_\lambda^{n+1} &= \left\{ \begin{array}{l} \mathbf{X}_\lambda^n + \Delta t \varphi_y \left(\mathbf{X}_\lambda^n, t^n\right) \\ + \dfrac{\Delta t^2}{2} \left(\begin{array}{r} \varphi_{yt} \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_x \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_{yx} \left(\mathbf{X}_\lambda^n, t^n\right) \\ + \varphi_y \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_{yy} \left(\mathbf{X}_\lambda^n, t^n\right) \end{array} \right) \end{array} \right\} \\
\frac{\mathbf{U}_\lambda^{n+1} - \mathbf{U}_\lambda^n}{\Delta t} &= \left\{ \begin{array}{l} \dfrac{1}{2} \left(\begin{array}{r} \varphi_{xt} \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_x \left(\mathbf{X}_\lambda^n, t^n\right) \cdot \varphi_{xx} \left(\mathbf{X}_\lambda^n, t^n\right) \\ + \varphi_y \left(\mathbf{X}_\lambda^n, t^n\right) \cdot \varphi_{xy} \left(\mathbf{X}_\lambda^n, t^n\right) \end{array} \right) \\ + \dfrac{1}{2} \left(\begin{array}{r} \varphi_{xt} \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) + \varphi_x \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \cdot \varphi_{xx} \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \\ + \varphi_y \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \cdot \varphi_{xy} \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \end{array} \right) \end{array} \right\} \qquad (2.76) \\
\frac{\mathbf{V}_\lambda^{n+1} - \mathbf{V}_\lambda^n}{\Delta t} &= \left\{ \begin{array}{l} \dfrac{1}{2} \left(\begin{array}{r} \varphi_{yt} \left(\mathbf{X}_\lambda^n, t^n\right) + \varphi_y \left(\mathbf{X}_\lambda^n, t^n\right) \cdot \varphi_{yx} \left(\mathbf{X}_\lambda^n, t^n\right) \\ + \varphi_y \left(\mathbf{X}_\lambda^n, t^n\right) \cdot \varphi_{yy} \left(\mathbf{X}_\lambda^n, t^n\right) \end{array} \right) \\ + \dfrac{1}{2} \left(\begin{array}{r} \varphi_{yt} \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) + \varphi_y \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \cdot \varphi_{yx} \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \\ + \varphi_y \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \cdot \varphi_{yy} \left(\mathbf{X}_\lambda^{n+1}, t^{n+1}\right) \end{array} \right) \end{array} \right\}
\end{aligned}
$$

The velocity Verlet has been used to estimate the particle trajectory due to a linear wave with $A = 0.01\,m$, $H = 0.1\,m$, $L = 1.0\,m$, and $T = 1.07261\,s$ and with initial position $x_0 = 0.5\,m$ and $y_0 = -0.02\,m$. The trajectory has been estimated for several time steps ($\Delta t = T/10$, $\Delta t = T/20$, $\Delta t = T/40$, and $\Delta t = T/80$), using the analytical solution for the initial velocity. Figure 2.11 compares the analytical solution of the trajectory for one wave period with the numerical ones. Comparing the final position of the analytical solution and the numerical ones, an error estimate is obtained. We use this error to carry out a convergence analysis (Table 2.11).

Now, we will integrate the particle trajectories using the semi-Lagrangian approach given the acceleration field induced by a linear wave. The acceleration and velocities at the mesh nodes ($\mathbf{a}_b^0$ and $\mathbf{u}_b^0$) are initialized with the analytical values. The particle's initial

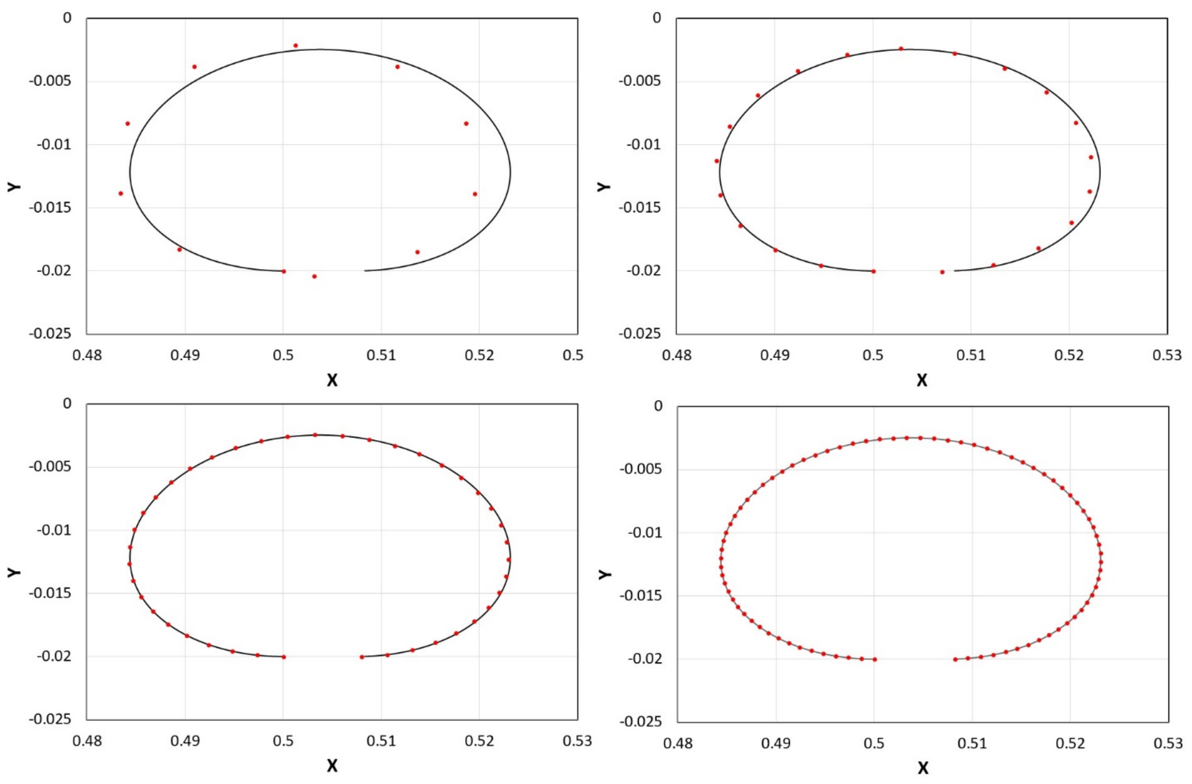


Fig. 2.11 Particle trajectory integration in the wave problem using the velocity Verlet algorithm. Black solid line represents the analytical solution. Red dots represent de numerical solution for $\Delta t = T/10$, $\Delta t = T/20$, $\Delta t = T/40$ and $\Delta t = T/80$

	$x(T)$	$y(T)$	Error **X**
Analytical	0.508308	-0.020000	
$\Delta t = T/10$	0.503153	-0.020397	0.00517037
$\Delta t = T/20$	0.507039	-0.020091	0.00127241
$\Delta t = T/40$	0.507992	-0.020015	0.00031649
$\Delta t = T/80$	0.508229	-0.019997	$7.9029 \cdot 10^{-5}$

	$v_x(T)$	$v_y(T)$	Error **V**
Analytical	-0.09850702	-0.00238937	
$\Delta t = T/10$	-0.09850997	-0.00178868	0.00060070
$\Delta t = T/20$	-0.09851727	-0.00222848	0.00016122
$\Delta t = T/40$	-0.09851019	-0.00234849	0.00004100
$\Delta t = T/80$	-0.09850785	-0.00237911	0.00001029

Table 2.1 Values and errors for the position and velocity of a particle at $t = T$.

velocities and accelerations are interpolated from the mesh initial values ($\mathbf{U}_\lambda^0 = \mathbf{u}_h^0\left(\mathbf{X}_\lambda^0\right)$ and $\mathbf{A}_\lambda^0 = \mathbf{a}_h^0\left(\mathbf{X}_\lambda^0\right)$) . Afterwards, during the simulation the acceleration field is imposed on the mesh nodes and interpolated at the particles, while the velocities are obtained via the semi-Lagrangian approach. The algorithm reads as follows:

- Step 1: Move particles from $t = n\Delta t$ to $t = (n+1)\,\Delta t$:

$$\mathbf{X}_\lambda^{n+1} = \mathbf{X}_\lambda^n + \Delta t \mathbf{u}_h^n\left(\mathbf{X}_\lambda^n\right) + \frac{\Delta t^2}{2}\mathbf{a}_h^n\left(\mathbf{X}_\lambda^n\right) \tag{2.77}$$

- Step 2: Interpolate the particle's acceleration at the new position:

$$\mathbf{A}_\lambda^{n+1} = \mathbf{a}_h^{n+1}\left(\mathbf{X}_\lambda^{n+1}\right) \tag{2.78}$$

- Step 3: The new particle´s velocity is updated:

$$\mathbf{U}_\lambda^{n+1} = \mathbf{U}_\lambda^n + \frac{1}{2}\Delta t\left(\mathbf{A}_\lambda^n + \mathbf{A}_\lambda^{n+1}\right) \tag{2.79}$$

- Step 4: The particle´s velocities are projected onto the mesh to obtain the new velocity field on the background mesh:

$$\mathbf{u}_h^{n+1} = \wp_{\{\lambda\}}^{n+1}\left\{\mathbf{U}_\lambda^{n+1}\right\} \tag{2.80}$$

In order to achieve second order convergence, each step must be at least second order in time and space. Step 1 has already been demonstrated that is second order. Step 2 interpolates from the mesh values to the particles using the FE linear shape functions. This interpolation is second order as demonstrated in [Bec14]. Step 3 integrates the velocity in time with a Crank-Nicolson scheme, which is known to be second order in time. And finally, Step 4 is the projection step using the global least square projector which has been demonstrated to be second order in space and time.

The time step has been set-up such that $h/\Delta t = 4.661$, being h the characteristic mesh size. Table 2.2 provides the values of the mean quadratic error (MQE) of the velocity on the mesh and Figure 2.12 shows that the rate of convergence is of second order, as expected.

	$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \left(\mathbf{u}_h\left(x_i\right) - \mathbf{u}_a\left(x_i\right)\right)^2}{N}}$
$h = H/2$	$1.226 \cdot 10^{-3}$
$h = H/4$	$2.345 \cdot 10^{-4}$
$h = H/6$	$1.176 \cdot 10^{-4}$
$h = H/8$	$6.951 \cdot 10^{-5}$
$h = H/10$	$4.742 \cdot 10^{-5}$

Table 2.2 Mean quadratic error of the velocity on the mesh at $t = T$

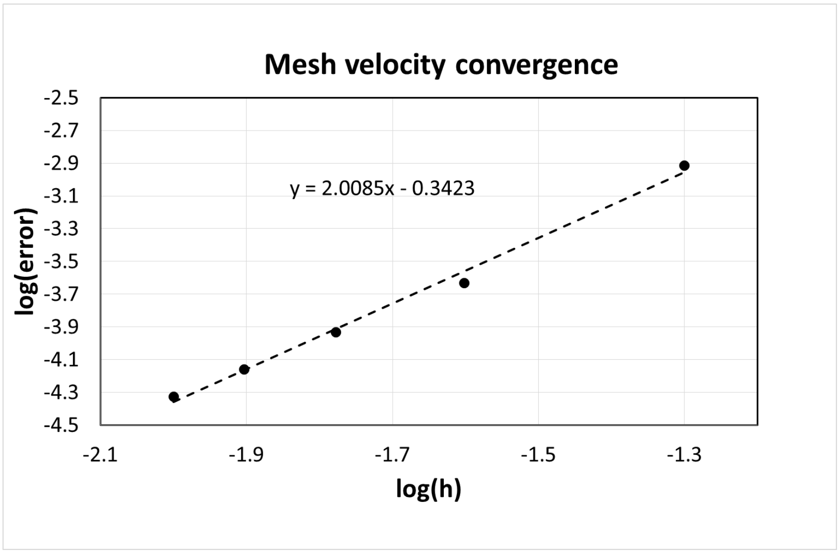


Fig. 2.12 Convergence analysis for the velocity on the mesh for a linear wave using the semi-Lagrangian approach.

2.6.3 2D Lid Driven Cavity

The particulars for this case study are given in table 2.3, and figure 2.13 shows the discretisation used and the boundary conditions applied. An average of 2 iterations is only required for convergence.

Lid Velocity $\mathbf{u}$	$-1\,m/s$
Reynolds number Re	1000
Domain size (x, y)	1m x 1m
Domain discretisation	80x80
Number of triangle elements	25600
Number of nodes	12961
Average number of particles per element	3
Mesh size $\Delta x = \Delta y$	0.0125
Time step Δt	0.1
Courant number	8
Simulation time	$10^5\Delta t$
Sampling time	$10^2\Delta t$

Table 2.3 Particulars of the lid driven cavity flow

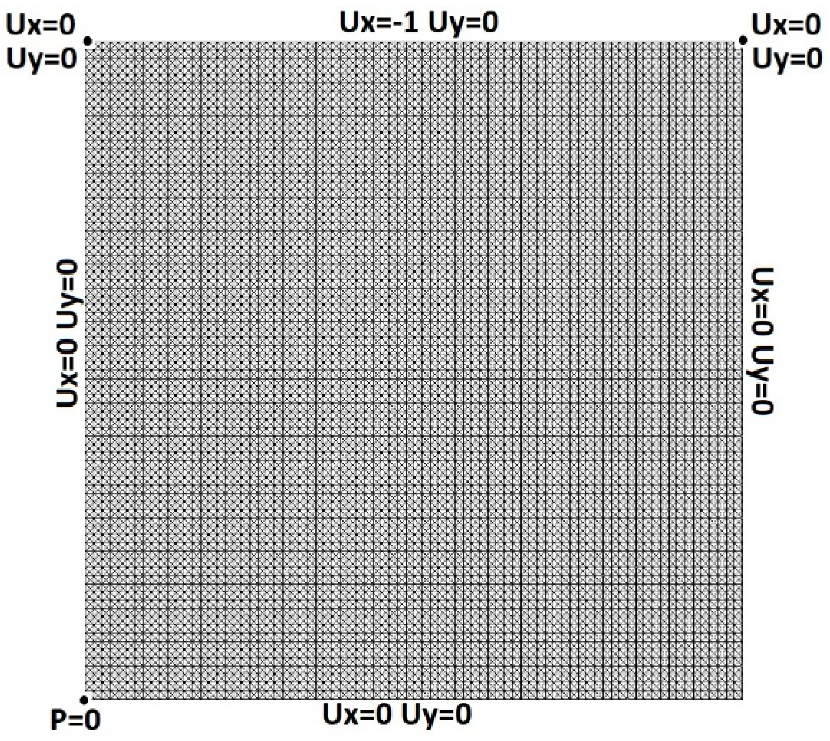
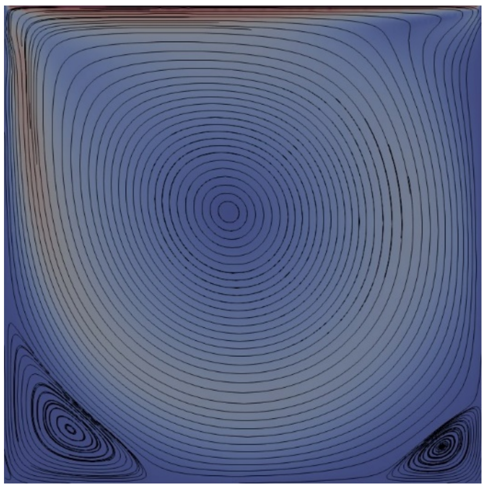


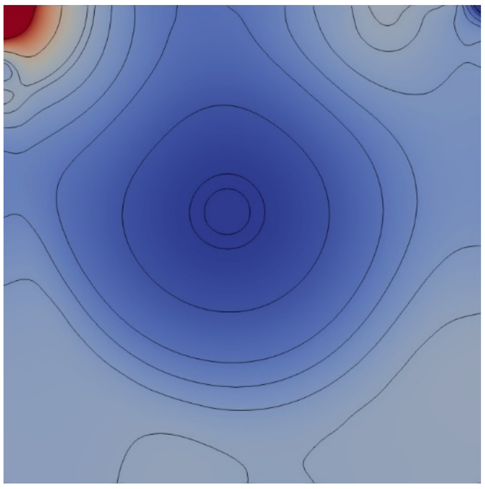
Fig. 2.13 Lid driven cavity mesh discretisation

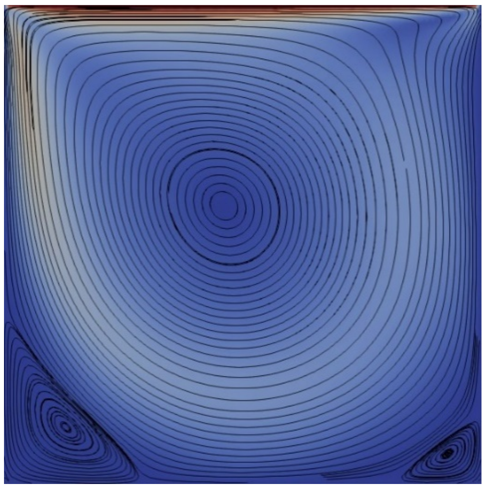
Although the steady state solution is achieved after one thousand time steps, the simulation time has been set to $T = 10^5\,\Delta t$ to verify whether the steady state solution shows symptoms of error accumulation.

The results obtained with the present second-order Verlet-SLPFEM are compared to those obtained using a high resolution spectral method in [BP98]. Also results are shown for a first-order SL-PFEM based on the Backward Euler integrator for the velocity equation. Figure 2.14 compares the streamlines and Figure 2.15 compares the horizontal velocity and pressure at the mid-section. While the second-order SL-PFEM based on the Verlet algorithm provides very similar results to those of [BP98], the first-order SL-PFEM based on the Euler scheme produces a highly diffusive solution on the pressure and velocity. Despite being a steady-state problem the first-order method is incapable of achieving the right solution with the current time step. It would require a significant reduction of the time step, which leads to a significant increase of CPU time. On the other hand, the second-order scheme produces an accurate solution for the current time step.

Figure 2.16 shows the evolution of the horizontal velocity at the node located at $(x, y) = (0.5, 0.175)$. It is observed that an average of 0.392 meters per second is obtained with a variance of $0.001 m/s$ approximately. No symptoms of error accumulation are observed.


(a) Velocity Streamlines using second order Verlet


(b) Pressure Contour using second order Verlet


(c) Velocity Streamlines using first order Euler

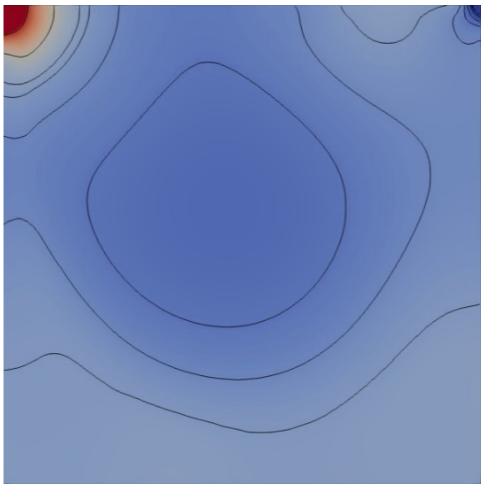
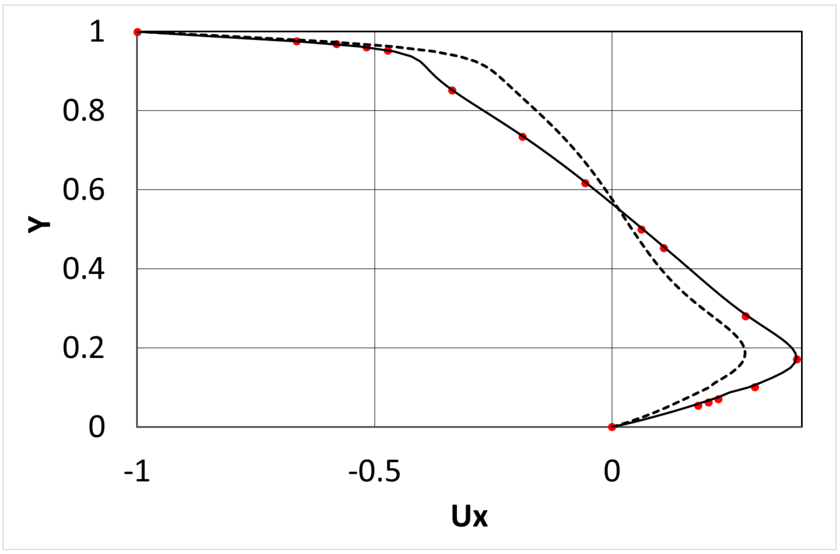
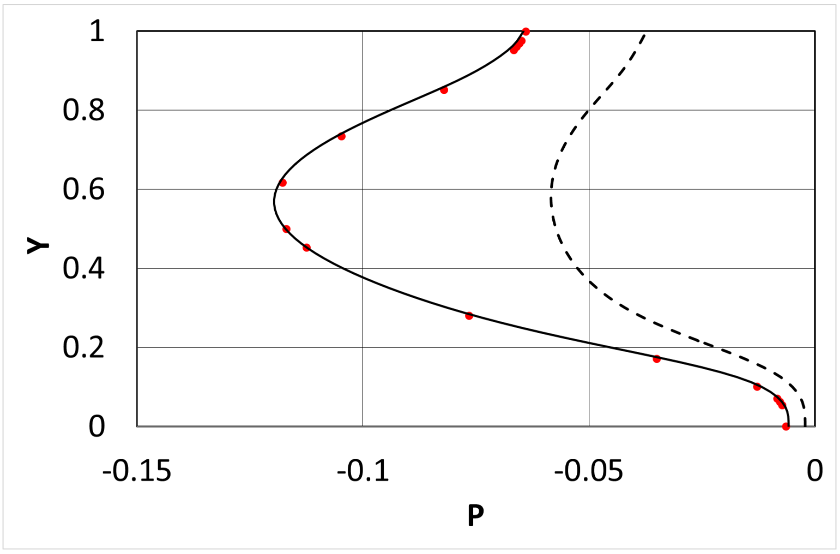

(d) Pressure Contour using first order Euler

Fig. 2.14 Velocity Streamlines and Pressure contour comparison



(a) Velocity Profile



(b) Pressure Profile

Fig. 2.15 Profile comparison at the midsection $x = 0.5$. Red dots are from the spectral method [BP98], the solid line the second order Verlet SL-PFEM and, the dash-line the first order Euler SL-PFEM.

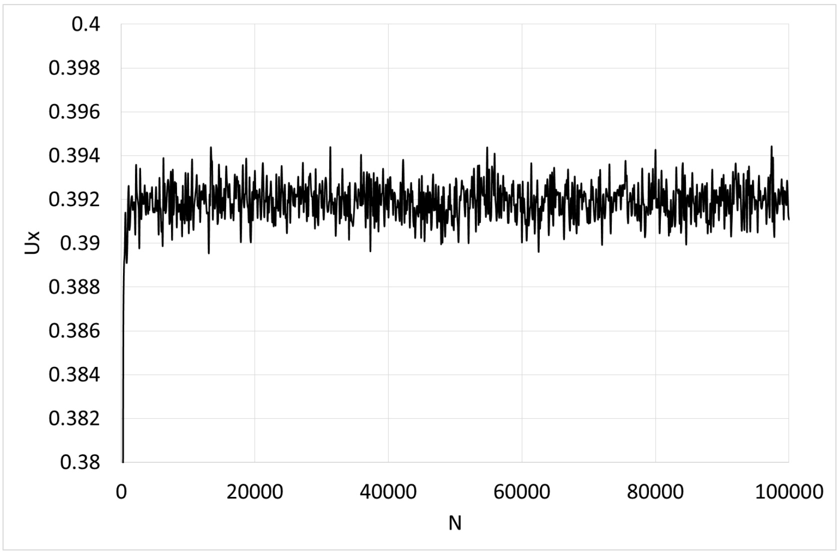


Fig. 2.16 Horizontal velocity evolution at $(x, y) = (0.5, 0.175)$ for the second order Verlet SL-PFEM

	Case 1	Case 2
Number of Elements	4096	65536
Number of Nodes	2113	33025
Number of Particles	12288	196608
Courant Number	2.54	10.1
Reynolds Number	2000	2000
Number of Time-Steps	500	500
Simulation Time	$250\,s$	$250\,s$

Table 2.4 Taylor Green Vortex case data

2.6.4 Two-dimensional Taylor-Green vortex

The two-dimensional Taylor-Green vortex problem is used to evaluate the convergence order of the numerical strategy solving an incompressible flow problem for which an analytical solution is known. The analytical solution is given by:

$$\begin{aligned}\mathbf{u}_x\left(x,y,t\right) &= -sin\left(x\right)\,cos\left(y\right)\,e^{-2\nu\,t}\\ \mathbf{u}_y\left(x,y,t\right) &= cos\left(x\right)\,sin\left(y\right)\,e^{-2\nu\,t}\\ P\left(x,y,t\right) &= \frac{1}{4}\left[cos\left(2x\right)+cos\left(2y\right)\right]e^{-4\nu\,t}\end{aligned} \tag{2.81}$$

Where $\nu = 0.001m^2/s$ is the kinematic viscosity. The problem is solved using a square domain $\left[-\pi,\pi\right]x\left[-\pi,\pi\right]$ with velocity periodic boundary conditions at the wall. Also, to prevent the pressure to be undetermined, the following condition is imposed at $(x,y) = (0,0)$:

$$P\left(0,0,t\right) = \frac{1}{2}e^{-4\nu\,t} \tag{2.82}$$

The known problem with this case when using periodic conditions is that the solution obtained is unsteady and is therefore unusable when performing a convergence test as the vortexes degenerate. It is worth showing the results, which compare as expected.

The velocity and pressure field are initialized at $t = 0$ with the analytical solution. The simulation is carried out until a final time of $t = 250s$ is reached allowing to observe the degeneration of the vortexes. Table 2.4 provides the information of the two cases analysed. Figure 2.17 shows the meshes used for both cases. Figure 2.18 presents the initial analytical solution for the first case and figure 2.19 and 2.20 presents the degenerated vortexes after 250 seconds.

For both cases, the figures show how the vortexes created becomes unstable. Therefore, in order to be able to perform convergence analysis another type of condition has to be applied instead of periodic velocity conditions. The idea is to force the vortexes to not degenerate.

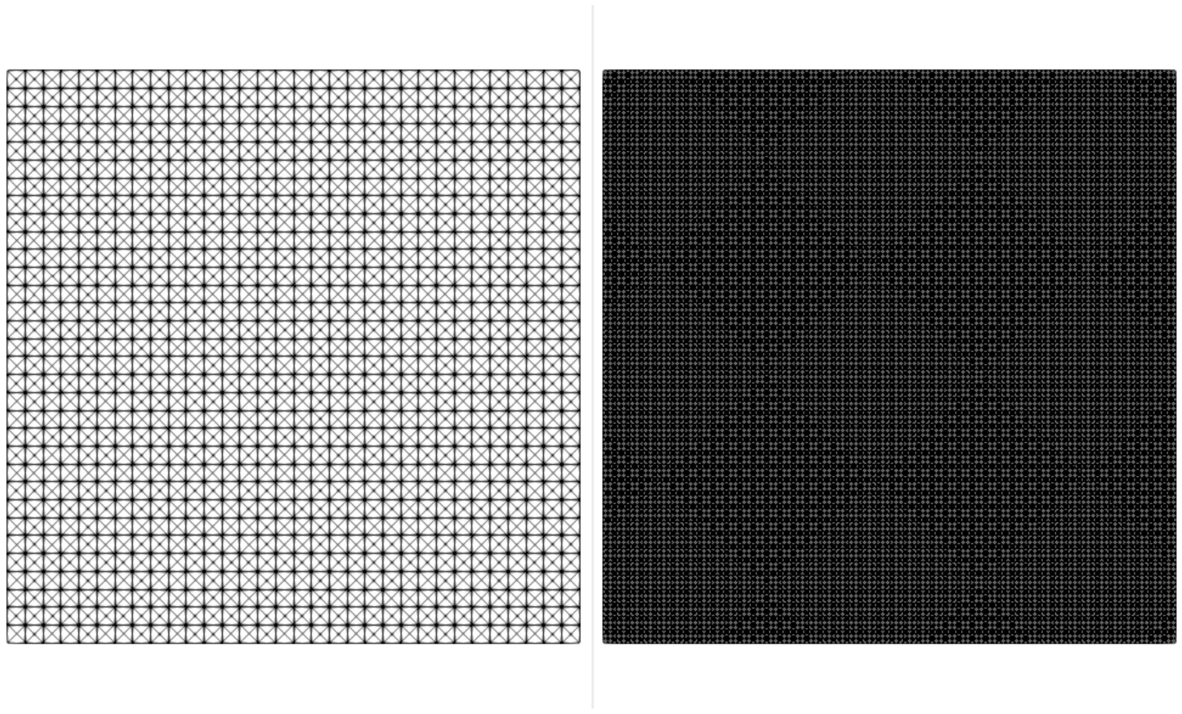


Fig. 2.17 Meshes used for the two cases. Left case using a 32×32 structured symmetric mesh, right case using 128×128 structured symmetric mesh.

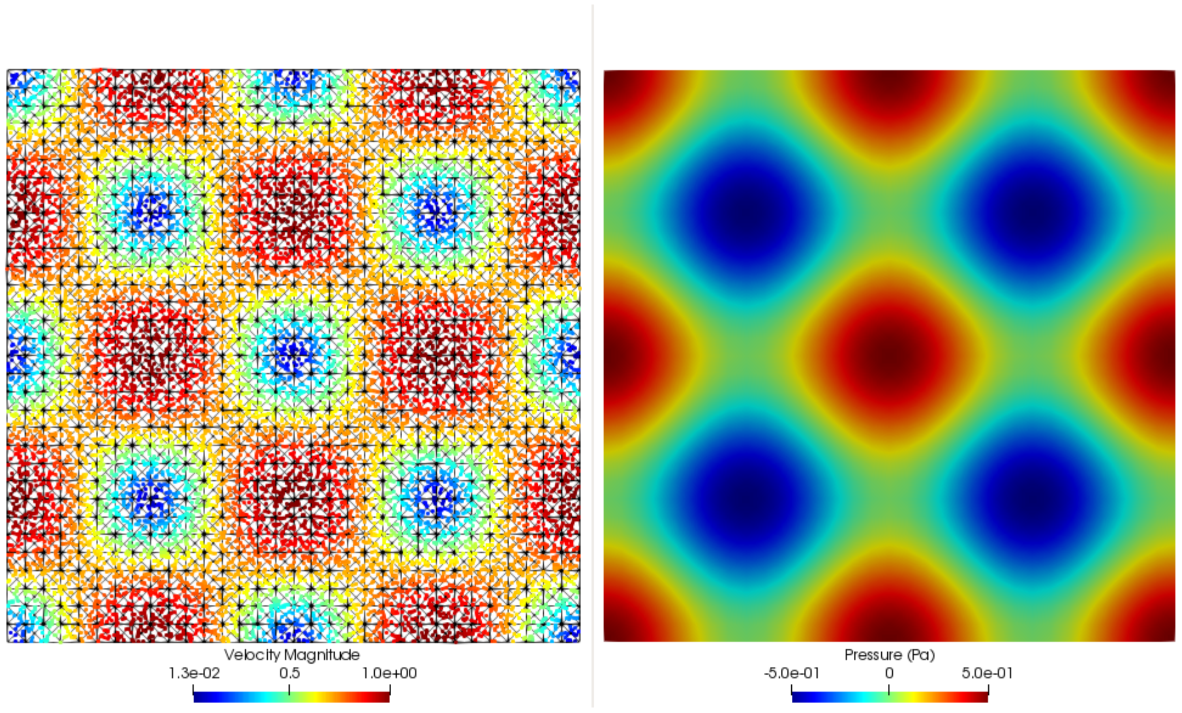


Fig. 2.18 Initial analytical result for the 32×32 mesh. Left: Velocity magnitude over the particles. Right: Pressure over the mesh.

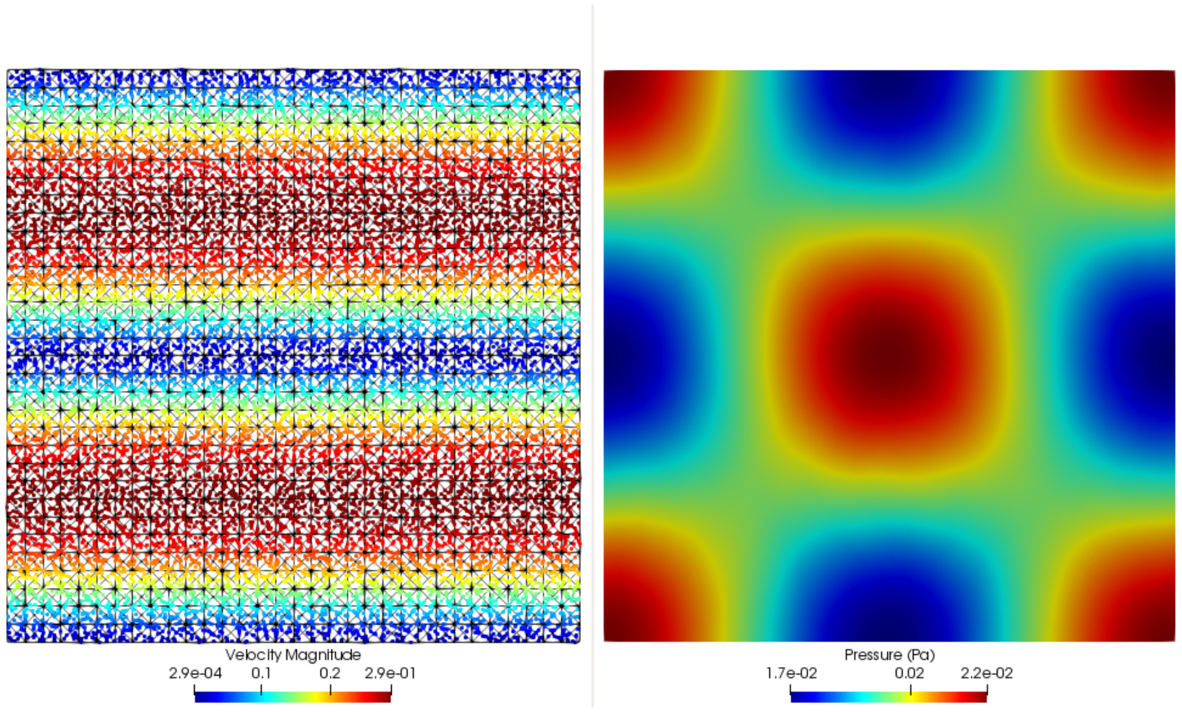


Fig. 2.19 Final result for the 32×32 mesh. Left: Velocity magnitude over the particles. Right: Pressure over the mesh.

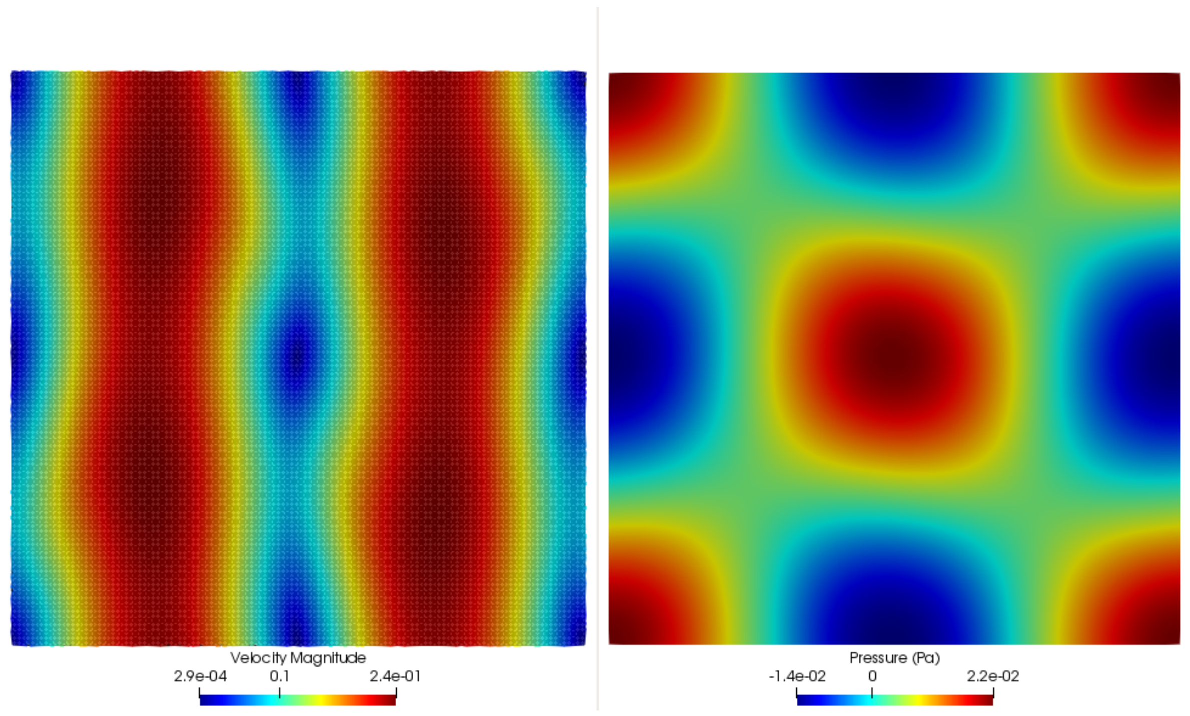


Fig. 2.20 Final result for the 128×128 mesh. Left: Velocity magnitude over the particles. Right: Pressure over the mesh.

2.6.5 Two-dimensional limited Taylor-Green vortex

The last validation showed how the periodic boundary conditions applied to the Taylor-Green vortex case are not adequate to perform the convergence analysis. In this case, velocity slip boundary conditions are applied to the walls and only one vortex is analysed therefore limiting the Taylor-Green vortex problem. The simulation is again started with the analytical solution shown at equation2.81.

Where $\nu = 0.001 m^2/s$ is the kinematic viscosity. This time, the domain is set using a square domain $[0,\pi]\, x\, [0,\pi]$ (which contains only one vortex) and, again, to prevent the pressure to be undetermined, the following condition is imposed at $(x,y) = (0,0)$:

$$P\left(0,0,t\right) = \frac{1}{2} e^{-4\nu\, t} \tag{2.83}$$

The velocity and pressure field are initialized at $t = 0$ with the analytical solution. The simulation is carried out until a final time of $t = 10s$ is reached. Table 2.5 provides the values of the mesh size, time step, and the number of time steps computed. Figure 2.21 shows the mesh used for case 2.

Case	Mesh size	Time step	Number of time steps
1	$\Delta x = \pi/8$	$\Delta t = 0.2\, s$	50
2	$\Delta x = \pi/16$	$\Delta t = 0.1\, s$	100
3	$\Delta x = \pi/24$	$\Delta t = 0.0667\, s$	150
4	$\Delta x = \pi/32$	$\Delta t = 0.05\, s$	200
5	$\Delta x = \pi/48$	$\Delta t = 0.0333\, s$	300
6	$\Delta x = \pi/64$	$\Delta t = 0.025\, s$	400

Table 2.5 Numerical parameters used in the 2D Taylor-Green Vortex

Figure 2.22 shows the convergence of the proposed algorithm. It can be seen that both, pressure and velocity converge with second order convergence rate.

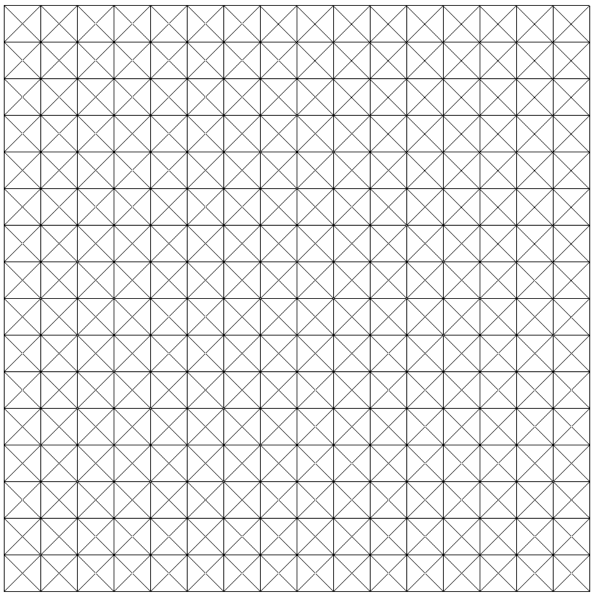


Fig. 2.21 Structured mesh with $\Delta x = \pi/16$

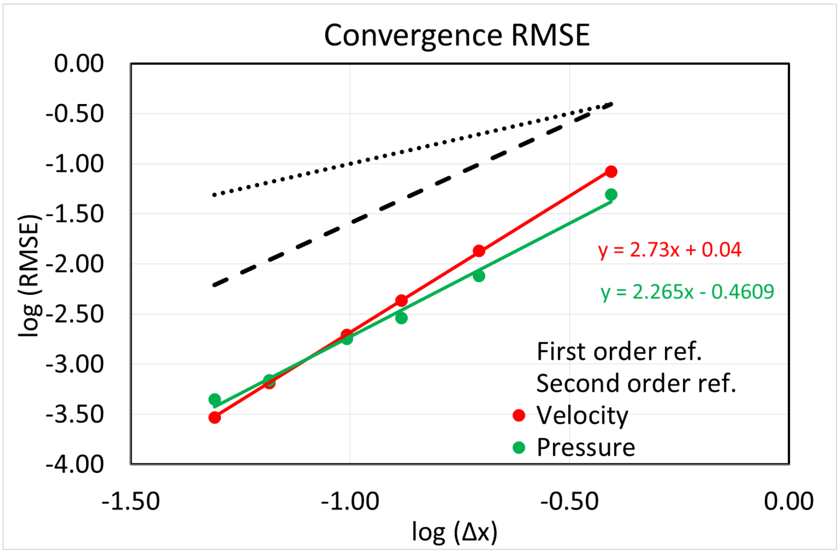


Fig. 2.22 Convergence analysis for the 2D Taylor-Green Vortex. Black dashed line represents first order convergence. Black dotted line represents second order convergence. Green: pressure errors. Red: velocity errors.

2.6.6 Two-dimensional Steady Taylor-Green vortex

A modified version of the two-dimensional Taylor-Green vortex problem is used to evaluate the convergence order of the numerical strategy solving an incompressible flow problem for which an analytical solution is known. In this version, a mass force is introduced to replace the energy dissipated by the viscous terms, so that a steady solution is obtained. The mass force to be introduced is:

$$\begin{aligned} f_x(x,y) &= -2\nu\, sin(x)\, cos(y) \\ f_y(x,y) &= 2\nu\, cos(x)\, sin(y) \end{aligned} \tag{2.84}$$

And the analytical solution to the problem is:

$$\begin{aligned} \mathbf{u}_x(x,y) &= -sin(x)\, cos(y) \\ \mathbf{u}_y(x,y) &= cos(x)\, sin(y) \\ P(x,y) &= \frac{1}{4}\left[cos(2x) + cos(2y)\right] \end{aligned} \tag{2.85}$$

The problem is solved with $\nu = 0.01 m^2/s$ $Re = 314$. The fluid domain is a square of $[0,\pi]\, x\, [0,\pi]$. Slip boundary conditions are used for the velocity. Also, to prevent the pressure to be undetermined, the following Dirichlet condition is imposed:

$$P(0,0) = 0.5\, Pa \tag{2.86}$$

The velocity and pressure field are initialized with the analytical solution. And average of 2 iterations is only required for convergence. The simulation is carried out for $t = 400s$. Based on the maximum velocity $\mathbf{u}_{max} = 1\, m/s$, the Courant number used is $Cr = \Delta t/\Delta x = 2.04$. The case is initiated with 3 particles per element, and particles are removed when it exceeds 6 particles per element. At the end of the simulation the average number of particles per element is slightly lower than 3.

Table 2.6 provides the values of the mesh size, time step, and the number of time steps computed. Figure 2.23 shows an intermediate mesh used in the analysis. Figure 2.24 shows how the pressure contours on the mesh and the particle's velocity obtained for case 64. Both results are quite close to the analytical solution. For this case, the root mean square error is in the order of 0.003 for both, velocity and pressure.

Case	Mesh size	Time step	Number of time steps
16	$\Delta x = \pi/16$	$\Delta t = 0.4\,s$	1000
24	$\Delta x = \pi/24$	$\Delta t = 0.26667\,s$	1500
32	$\Delta x = \pi/32$	$\Delta t = 0.2\,s$	2000
48	$\Delta x = \pi/48$	$\Delta t = 0.13333\,s$	3000
64	$\Delta x = \pi/64$	$\Delta t = 0.1\,s$	4000
96	$\Delta x = \pi/96$	$\Delta t = 0.06667\,s$	6000
128	$\Delta x = \pi/128$	$\Delta t = 0.05\,s$	8000
192	$\Delta x = \pi/192$	$\Delta t = 0.03333\,s$	12000

Table 2.6 Numerical parameters use in the 2D Steady Taylor-Green Vortex

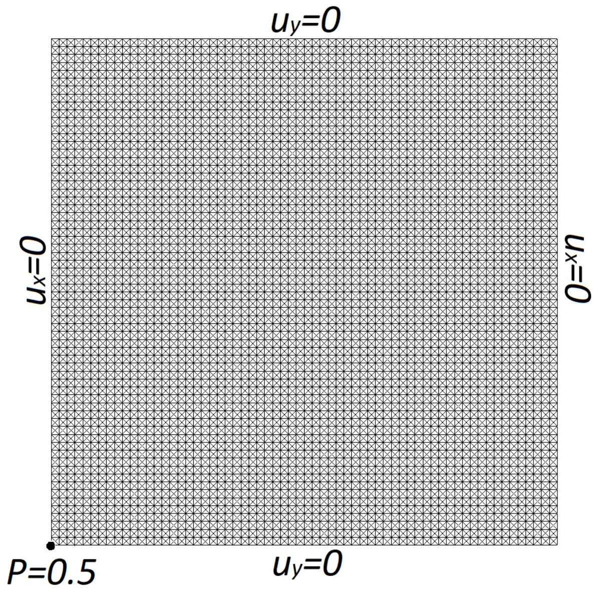
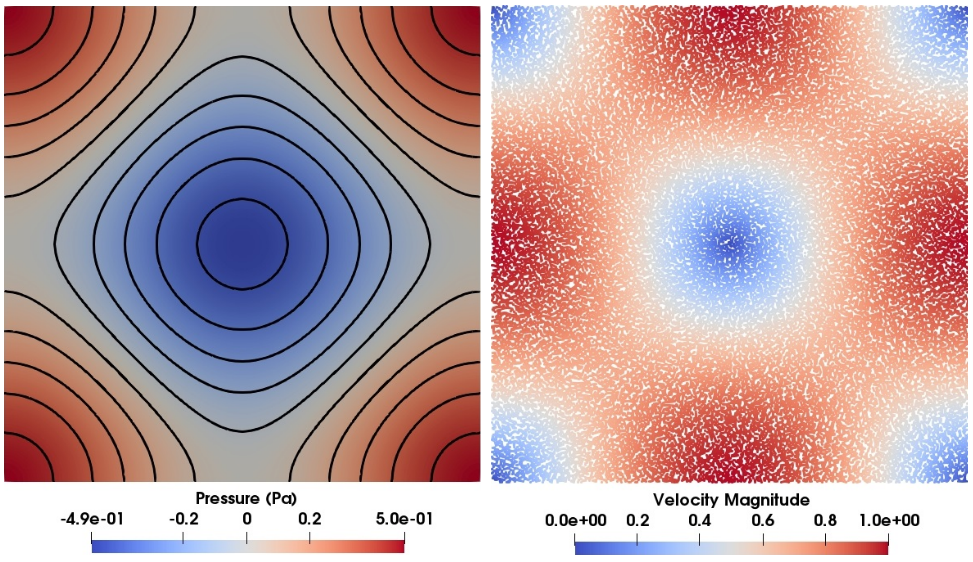


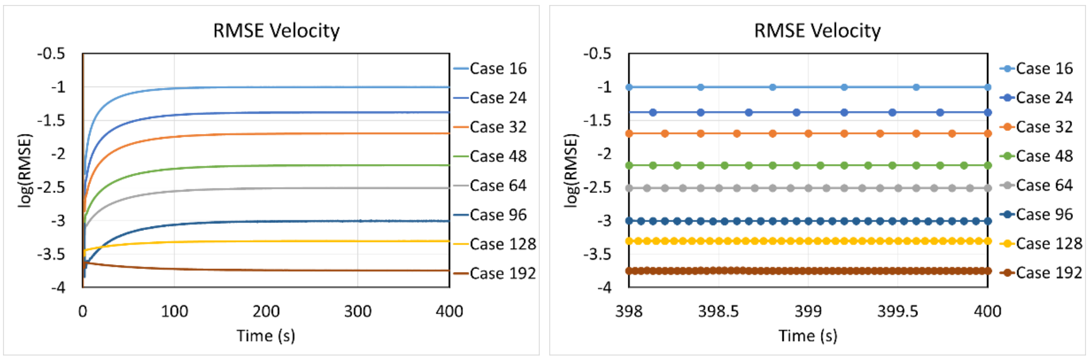
Fig. 2.23 Structured mesh with $\Delta x = \pi/64$ and boundary conditions applied



(a) Pressure contours (b) Particle velocity magnitude

Fig. 2.24 Results obtained from case 64.

Figure 2.25 shows the evolution of the errors along the simulation. It is observed that the errors achieved a constant value. Hence no symptoms of error accumulation over time is appreciated. Figure 2.26 provides the rate of convergence obtained using an average of three particles per element. The root mean square error (RMSE) has been used for the analysis. Table 2.7 shows the convergence of the proposed algorithm for pressure and velocity.



(a) Velocity RMSE over the full simulation time (b) Velocity RMSE at the last 2 seconds of simulation time

Fig. 2.25 Error evolution along time.

While in an Eulerian method all dependent variables remain unchanged once the steady state solution is reached, this is not the case in the SL-PFEM method. From the point of view of the particles, the flow is never steady since the intrinsic variables they transport are continually changing.

Convergence Rate	
Velocity	Pressure
2.59	2.66

Table 2.7 Rate of convergence.

If the chosen SL-PFEM scheme is not properly tailored, then the error accumulation will make the method to be incapable of computing a steady-state solution. This is because the accumulation of errors along the simulation time will eventually cause a numerical blow up. This case shows how the chosen SL-PFEM formulation for this thesis is capable of computing steady state solutions without errors concerns.

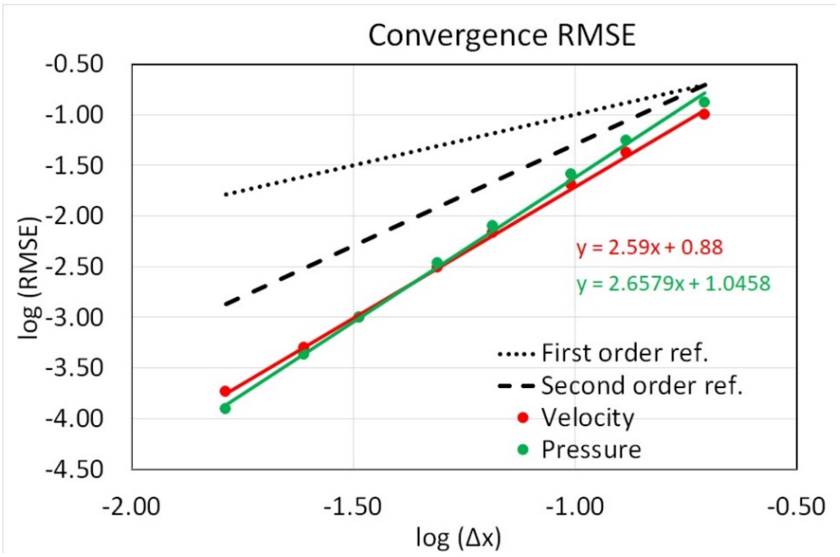


Fig. 2.26 Convergence analysis for the 2D Taylor-Green Vortex. Black dashed line represents first order convergence. Black dotted line represents second order convergence. Blue: pressure errors. Red: velocity errors

2.6.7 2D Flow around a cylinder - Von Kármán vortex street

This case studies the vortex development behind a circular cylinder, at Reynolds numbers from 100 to 1000, for which we expect a vortex street in the wake of the cylinder, the well known von Kármán vortex street. The case particulars are presented at table 2.8 and figure 2.27 presents the domain used for the problem.

The fluid domain is a two dimensional geometry with a cylinder of radius $R = 0.5\,m$. The obtained numerical results are compared to the experimental ones presented at [JC17; RD53].

Cylinder diameter	$1\,m$
Inlet velocity	$1\,m/s$
Fluid Density	$1\,Kg/m^3$
Fix velocity condition	$\Gamma_{Wall/Bodies}$
Fix X and Y velocity field	Γ_{Inlet}
Fix pressure field	Γ_{Outlet}
Fix Y Velocity	$\Gamma_{Velocity}$

Table 2.8 2D flow around a cylinder problem properties.

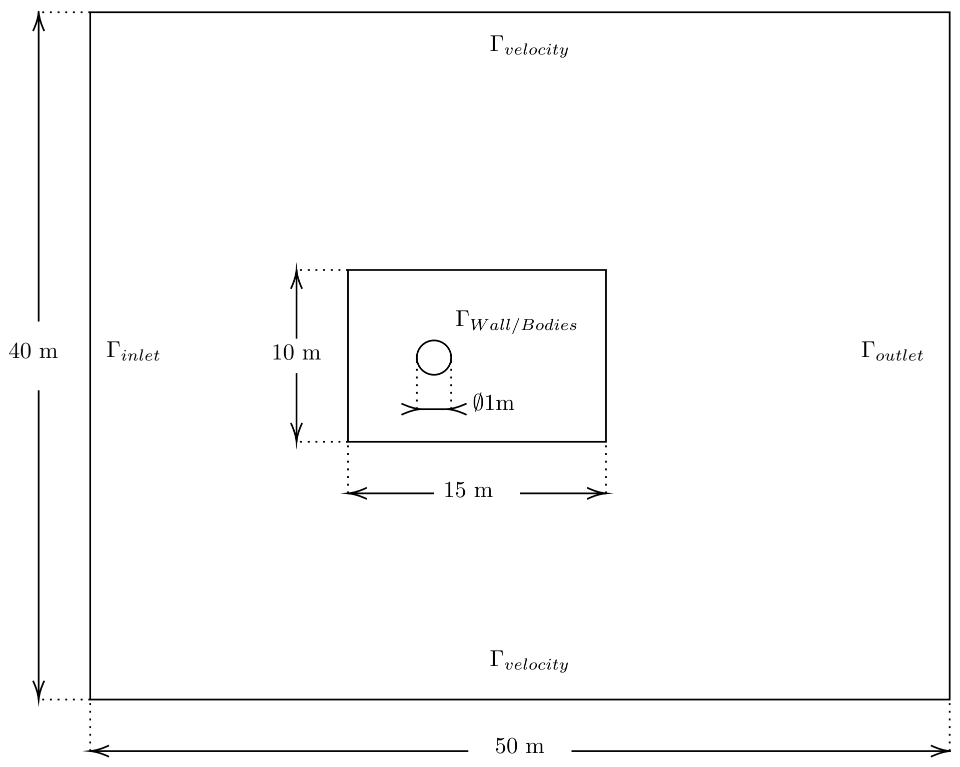
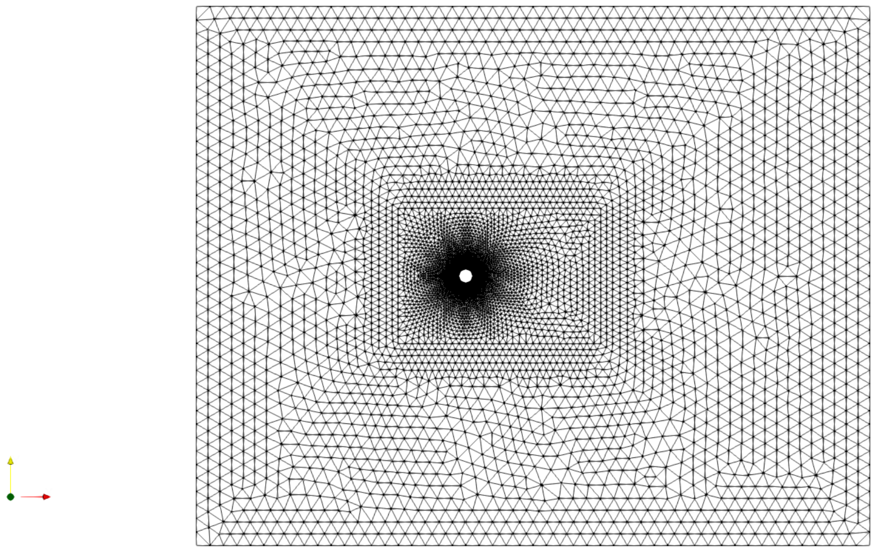


Fig. 2.27 Computational domain used to generate the structured mesh

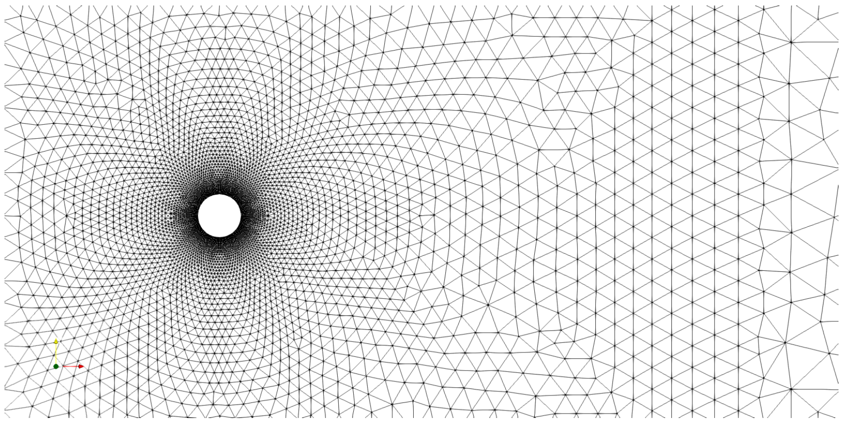
The proposed mesh is presented at 2.28 with a refinement zone surrounding the cylinder (inner domain) and with an even finer mesh at the contour of the cylinder. The sizes used are presented at table 2.9.

Zone	Mesh Size
Cylinder contour	$0.025\,m$
Inner domain	$0.5\,m$
Outer domain	$1.0\,m$

Table 2.9 2D flow around a cylinder mesh sizes.



(a) Mesh of the whole domain.



(b) Close up of the cylinder.

Fig. 2.28 Mesh used for the 2D flow around the cylinder.

Fluid characteristics are changed in order to obtain different reynolds numbers for each case. Table 2.10 presents the different cases so where it can be seen that different viscosities are used to obtain the reynold number of each case allowing to keep other properties constant for the sake of simplicity.

Re	$\rho\,[Kg/m^3]$	$V\ [m/s]$	$R\ [m]$	$\nu\ [m^2/s]$
100	1.0	1.0	0.5	0.01
200	1.0	1.0	0.5	0.005
300	1.0	1.0	0.5	0.00333
400	1.0	1.0	0.5	0.0025
600	1.0	1.0	0.5	0.00167
800	1.0	1.0	0.5	0.00125
1000	1.0	1.0	0.5	0.001

Table 2.10 Physical parameters used for the different cases.

The frequency of the vortex shedding in a Kármán vortex street is a characteristic of the flow behind a circular cylinder, and has been extensively measured and reported in references [JC17; RD53; SG17]. A regular Kármán street can be observed in the range $60 < \text{Re} < 5000$. At lower Reynolds numbers the wake does not oscillate, while at higher Reynolds numbers turbulent mixing appears.

The shedding frequency is usually expressed in terms of the dimensionless frequency, known as Strouhal number S, as follows:

$$S = \frac{f \cdot D}{V} \tag{2.87}$$

where f is the shedding frequency (from one side of the cylinder), D is the cylinder diameter, and V is the free-stream velocity. It should be noted that the Strouhal number depends only on the Reynolds number.

Therefore, figure 2.29 presents the different strouhal numbers obtained for different Reynold numbers and compared with the experimental results obtained at [JC17]. As it can be seen good agreement is obtained.

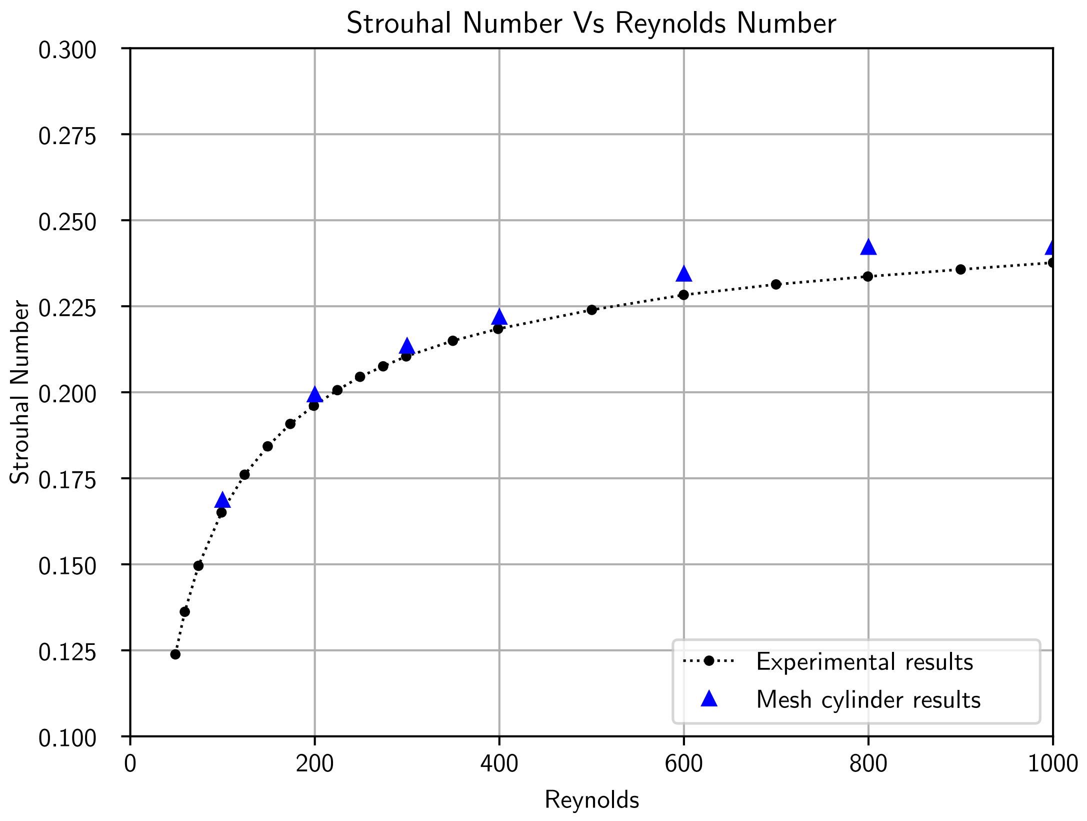


Fig. 2.29 Strouhal numbers obtained numerically compared with the experimental results from [JC17]

2.6.8 3D Flow around a cylinder - Convergence analysis

The three dimensional flow around a cylinder is simulated to assess the suitability of the present method for solving three dimensional problems. The particulars of the problem are given in Table 2.11. An average of $2-3$ iterations are only required for convergence.

Cylinder diameter (D)	$1\,m$
Cylinder height (H)	$1\,m$
Inlet Velocity (V)	$1\,m/s$
Viscosity (ν)	0.005
Re	200

Table 2.11 Flow past a cylinder particulars

For a Reynolds number of 200 and an aspect ratio of the cylinder height (H) to the cylinder diameter (D) of $H/D=1$, the flow behaves as a two dimensional flow [JC17]. A three dimensional structured mesh with five levels of refinement has been used, keeping the Courant number constant. Figure 2.30 shows the computational domain used, and Figure 2.31 shows the corresponding meshes used in this analysis. The particulars for each mesh are provided in Table 2.12, as well as the total number of time steps simulated. In average, two particles per elements are used.

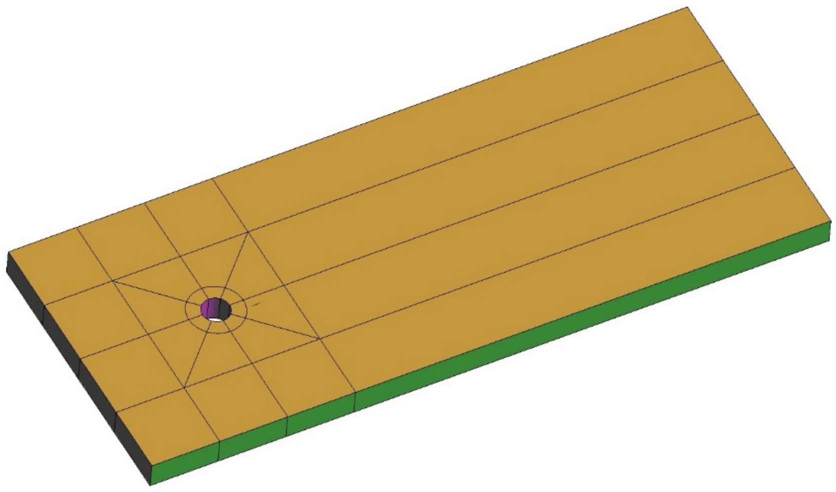


Fig. 2.30 Computational domain used to generate the structured mesh

Number of
tetras

Table 2.13 provides the Strouhal numbers obtained in each case study. It is observed how the lift force amplitude and the Strouhal number converge towards values of 0.27

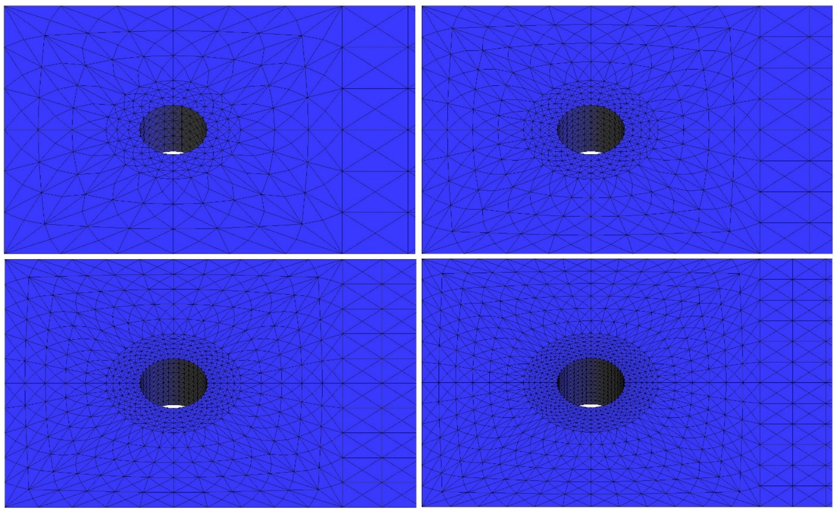


Fig. 2.31 Computational mesh refinement for the 3D flow past a cylinder

Mesh	Δx	Δt	Number of tetras	Number of Nodes	Number of Time-steps
1	0.1333	0.0667	62208	14487	1500
2	0.1	0.05	147456	33412	2000
3	0.08	0.04	288000	64185	2500
4	0.0667	0.0333	497664	109686	3000

Table 2.12 Numerical data used in the 3D flow past a cylinder

and 0.196, respectively. Figure 2.32 plots the logarithmic errors for the Strouhal number versus the logarithmic mesh size for each case study. Comparing with the second order reference line, it is observed that the convergence rate of the Strouhal number is close to second order. Figure 2.33 and Figure 2.34 provides a snapshot of the pressure field, particles velocities, streamlines and vorticity for Case 2.

Case	St	Error St	C_L
1	0.133	0.0630	0.177
2	0.162	0.0339	0.255
3	0.178	0.0184	0.246
4	0.186	0.0101	0.258
Ref. [JC17]	0.196		

Table 2.13 Numerical results for the 3D flow past a cylinder

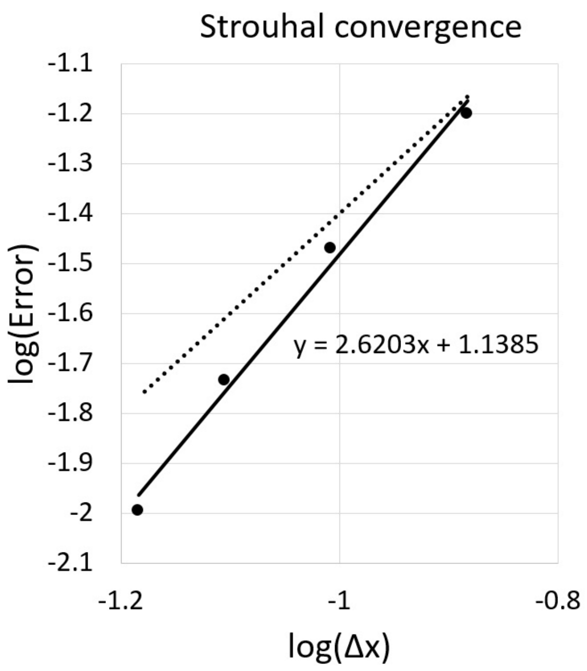
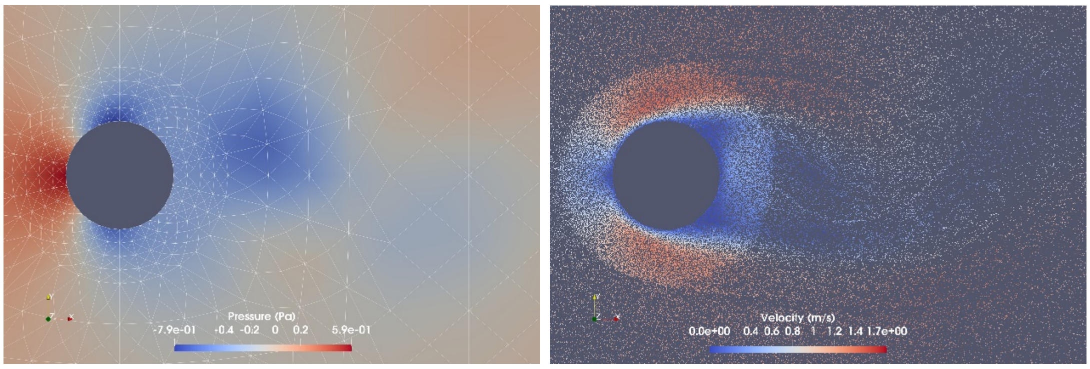


Fig. 2.32 Rate of convergence for the Strouhal number. Circles represent the log (error) for each case study. Solid line represents the trend of the convergence. Dot line represents second order convergence.


Pressure (Pa)
-7.9e-01 -0.4 -0.2 0 0.2 5.9e-01
Velocity (m/s)
0.0e+00 0.4 0.6 0.8 1 1.2 1.4 1.7e+00

(a) Pressure map on the mesh (b) Particle velocities

Fig. 2.33 Case 2 snapshots

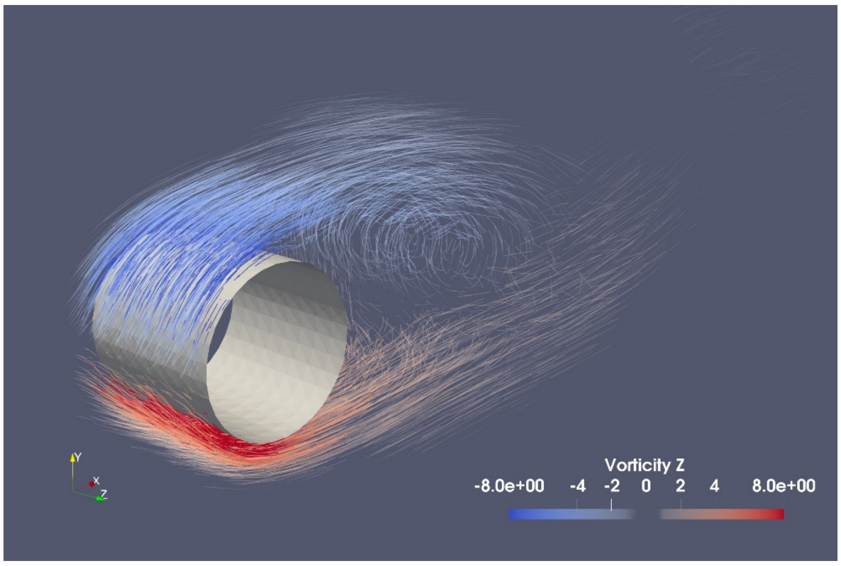


Fig. 2.34 Case 2 snapshots: oblique view of vorticity on particles

2.6.9 3D Flow around a cylinder - Vortex comparison

This verification case is a continuation of the cylinder validations. The same reference [JC17] not only presents results of the strouhal numbers or lift coefficients but it also presents the downstream vortex generation. So, here we will be comparing the results obtained using the current scheme with the one presented at the reference.

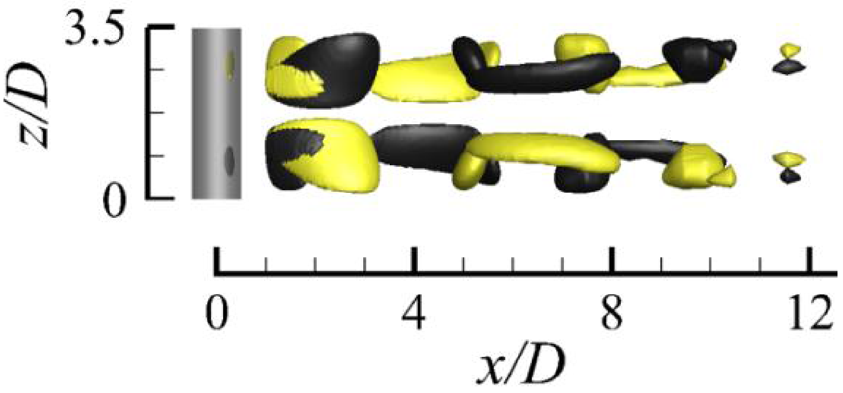


Fig. 2.35 Reference vorticity on X-axis (± 0.5) for the 3D cylinder flow ($z = 3.5\,D$)

In [JC17] DNS simulations are carried out for different cylinder lengths. Figure 2.35 presents the case we will be analysing and comparing against in this verification. The data of the cylinder and fluid flow are presented at table 2.14 and figure 2.36 presents the domain used which is the same one used for case 2.6.7 but in 3D and for a Reynolds number of $Re = 200$. The mesh used in this case is a bit more refined compared to 2.6.7 as here the idea is to capture the vortices and view them properly. Mesh data is presented at table 2.15. Finally, the applied conditions are the same as the ones at 2.6.7 being slip conditions at the domain walls, fix velocity conditions at the cylinder wall, fixed velocity at the inlet and fix pressure at the outlet.

Cylinder diameter	$1\,m$
Cylinder height	$3.5\,m$
Inlet velocity	$1\,m/s$
Fluid Density	$1\,Kg/m^3$
Fluid Viscosity	$0.005\,m^2/s$
Reynolds	200

Table 2.14 Data of the problem and fluid for the 3D flow around a cylinder

The results obtained are presented at figure 2.37 where the x-axis vorticity for values $\pm 0.5\,s^{-1}$ is presented and compared with the results from [JC17]. Good accordance in the obtained shapes is observed between the two results. This particular case was

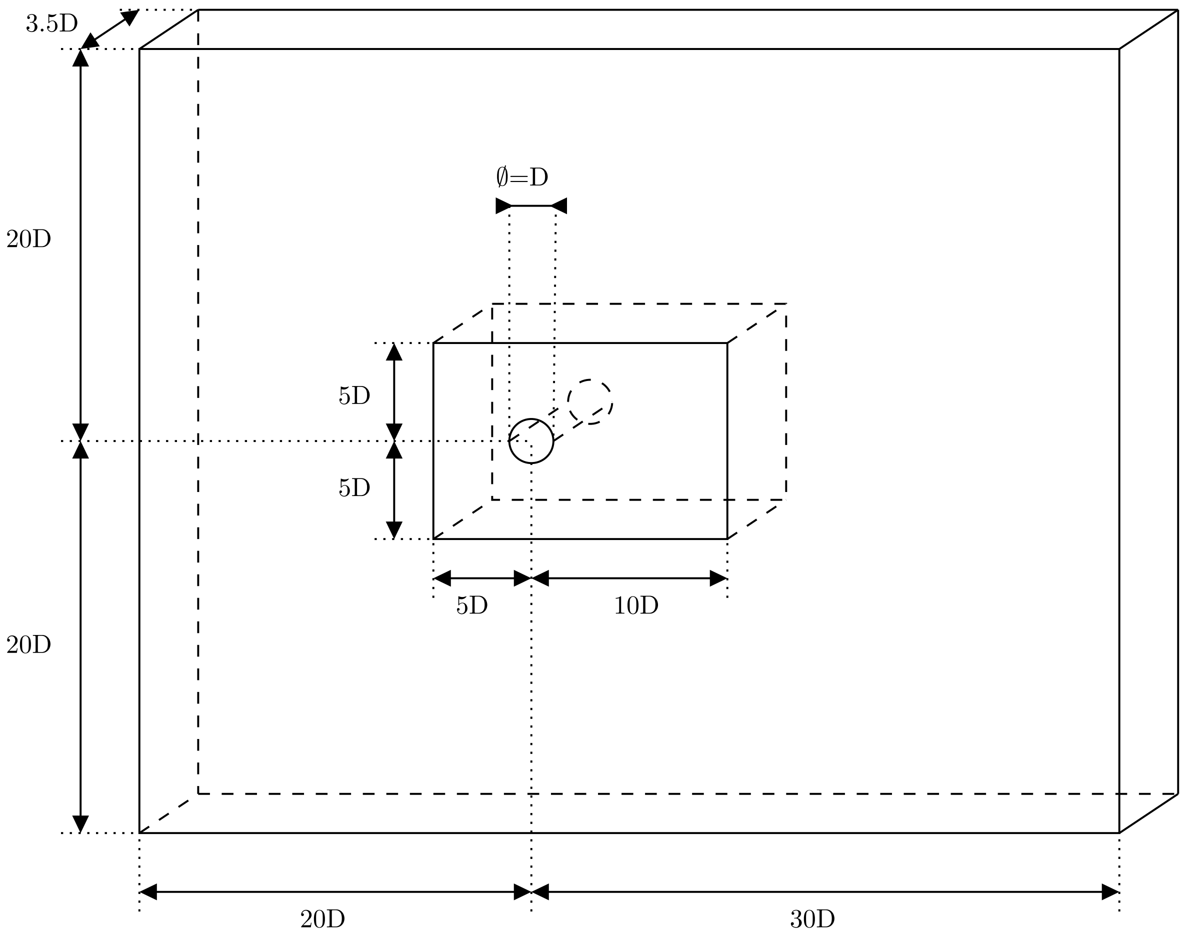


Fig. 2.36 Domain used for 3D cylinder flow ($z = 3.5\,D$).

Zone	Mesh size
Cylinder wall	$0.025D$
Inner Domain	$0.1D$
Outer Domain	$1.0D$

Table 2.15 Mesh data used for the vorticity comparison of the 3D flow around a cylinder.

chosen due to the sensibility needed in the scheme in order to properly catch the vortexes. This cylinder length is right between the 2D and 3D flow type (graph 2.38) as shown in the reference [JC17]. The transition between the two flow types produces a drop in the strouhal number and appears for cylinder lengths/Diameter ratios of $L/D > 3$. The obtained strouhal from this simulation is:

$$St = 0.184 \tag{2.88}$$

Which is in accordance with the drop observed at graph 2.38 where for the relation $L/D = 3.5$ the obtained strouhal is of $St = 0.189$.

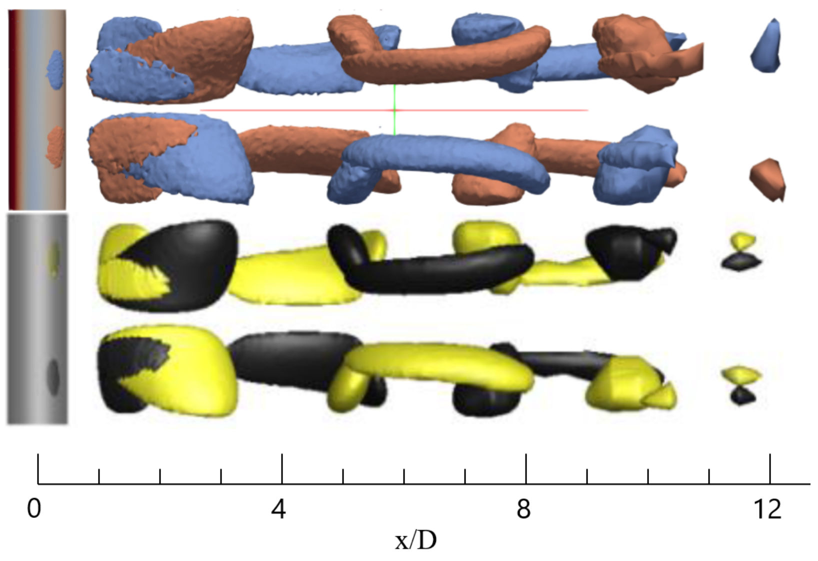


Fig. 2.37 Comparison of the X-Axis vorticity (± 0.5) of th 3D cylinder flow ($z = 3.5\,D$) for the current scheme (top) and the reference [JC17] (bottom)

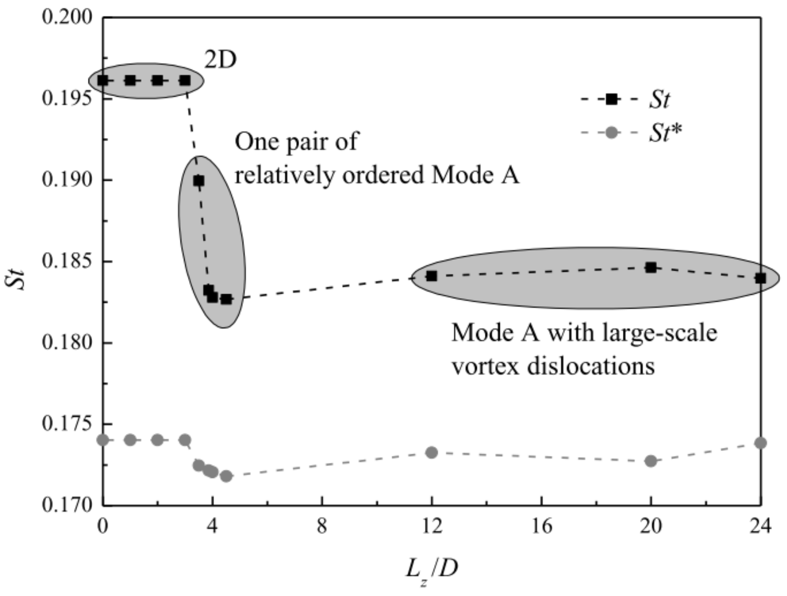


Fig. 2.38 Flow types: Variation of the strouhal number depending on the cylinder length.

2.6.10 3D flow - Wind resistance coefficient

This verification case will showcase the wind resistance of a ship with different heading angles obtained using the scheme proposed in this thesis compared with the wind resistance obtained using the regression lines shown at [FUI05] and [ITT14]. The references present regression lines that give the air resistance for different heading angles by using different parameters of the ship to analyse. It is worth mentioning that these regression lines have been obtained experimentally from a variety of vessels that have been submitted to different wind angles in a wind tunnel.

Data	Value
Overall Length	180.6 m
Beam	22.90 m
Depth	14.1 m
Displacement	14918 Metric Tons
Deadweight	7500 Metric Tons
Block Coefficient (Cb)	0.6
Installed power	11600 kW
Velocity at 85% MCR	18.0 kn
Autonomy	8000 $Miles$

Table 2.16 Data of the ship's model.

Fujiwara Data for Regression

A_{OD}	Lateral projected area of superstructures etc. on deck	450.6 m^2
A_{XV}	Area of maximum transverse section exposed to the winds	315.8 m^2
A_{YV}	Projected lateral area above the waterline	1337 m^2
b	Ship breadth	17.9 m
C_{MC}	Horizontal distance from midship section to centre of lateral projected area A_{YV}	0.9 m
h_{BR}	height of top of superstructure (bridge etc.)	17.5 m
h_C	Height from waterline to centre of lateral projected area A_{YV}	6.7 m
L_{OA}	Overall Length	116.6 m
μ	Smoothing range; normally 10(deg.)	0 deg
Ψ_{WR}	Relative wind direction; 0 means heading winds	180 deg

Table 2.17 Data used to obtain the regression line.

Therefore, in this verification an existing Ro-Ro ship, with parameters presented at 2.16 is used to obtain the numerical. In order to obtain the regression line, the parameters shown at figure 2.39 are presented at table 2.17.

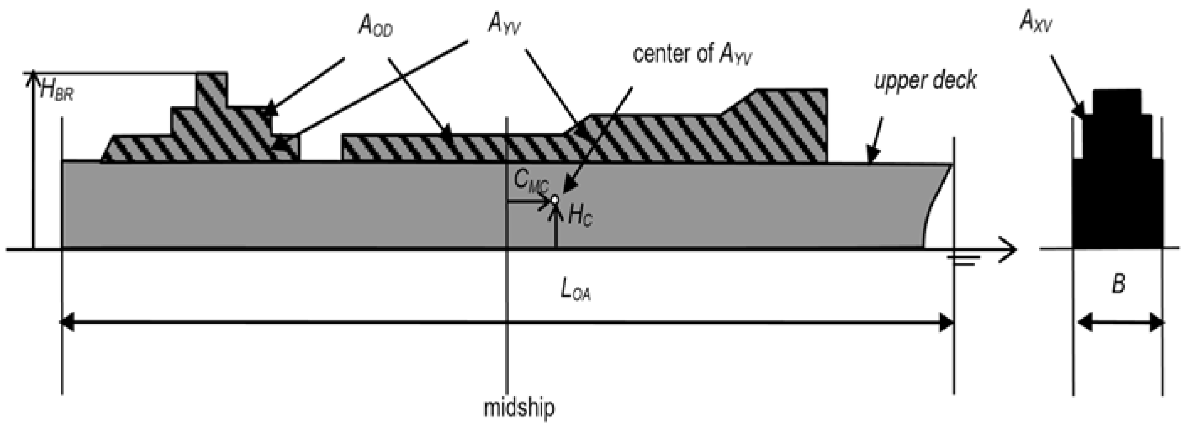


Fig. 2.39 Paramater nomenclature to use in order to obtain the air resistance regression line.

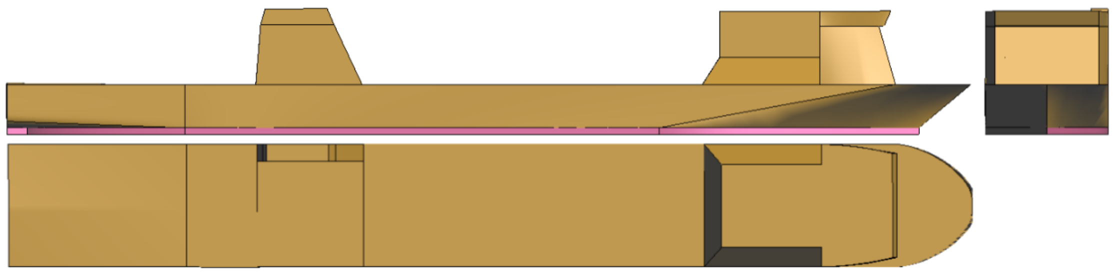


Fig. 2.40 Ro-RO model used for the numerical analysis

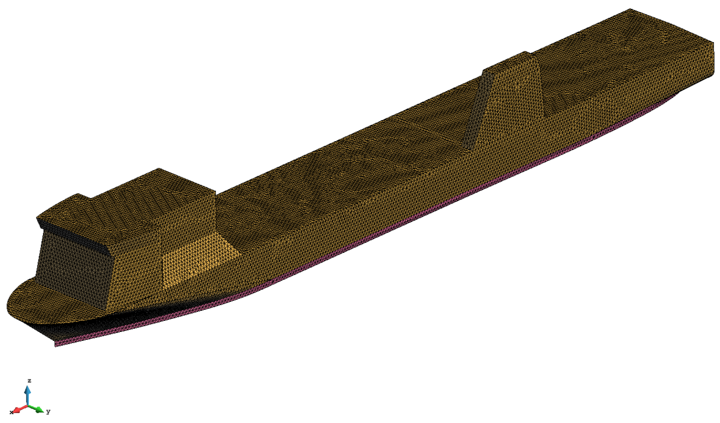


Fig. 2.41 Ro-RO 3D visualization with the surface mesh

Heading Angles	$0, 30, 60, 90, 120, 150, 180\, deg$
Fluid inlet velocity	$10\, m/s$
Fluid density	$1.17\, kg/m^3$
Fluid Viscosity	$1.8 \cdot 10^{-5}\, m^2/s$
Simulation time	$250\, s$
Time Step	$0.25\, s$

Table 2.18 Problem data used for the numerical simulation.

To model the problem, different cases are prepared where the ship is placed at different angles in order to obtain the resistance of the wind. Table 2.18 presents the different configurations and the fluid parameters. Furthermore, figures 2.40, 2.42 and 2.43 present the ship in question and the domain used to compute the simulations for 0 and 60 degree heading angle. Finally, the mesh data is presented at table 2.19 and figure 2.41 presents the mesh discretisation of the ship.

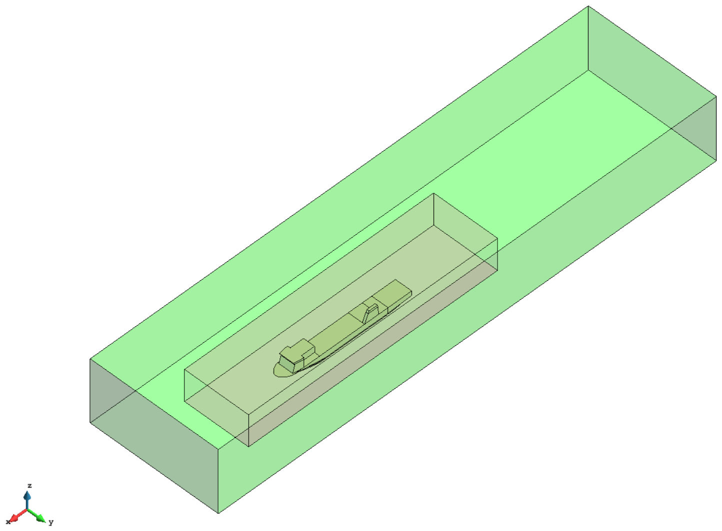


Fig. 2.42 Domain used for the case with 0 degree heading angle.

Ship Surface	$0.75\, m$
Inner domain	$4.0\, m$
Rest of domain	$8.0\, m$

Table 2.19 Mesh data used for the numerical simulation.

With all of this, the results obtained are summarised in one graph (figure 2.44) that shows the computed air resistance of the ship at the different angles and the obtained regression line using Fujiwara's method. As it can be observed, good agreement is obtained with the

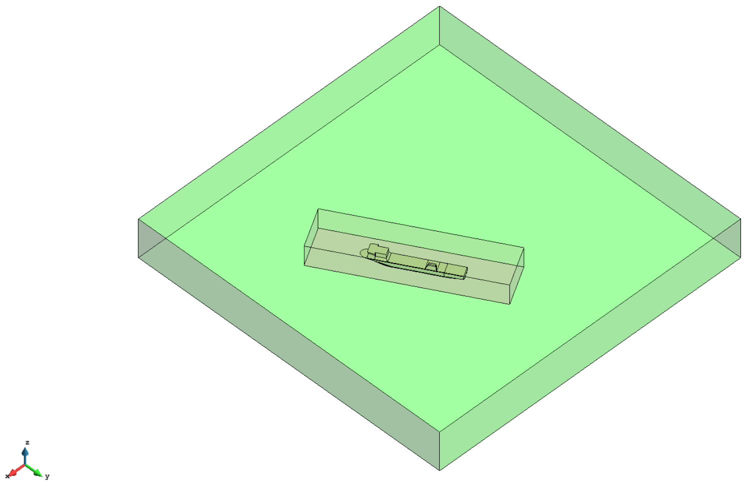


Fig. 2.43 Domain used for the case with 60 degree heading angle.

regression line. The following figures 2.45, 2.46 show the fluid flow obtained numerically for different heading angles.

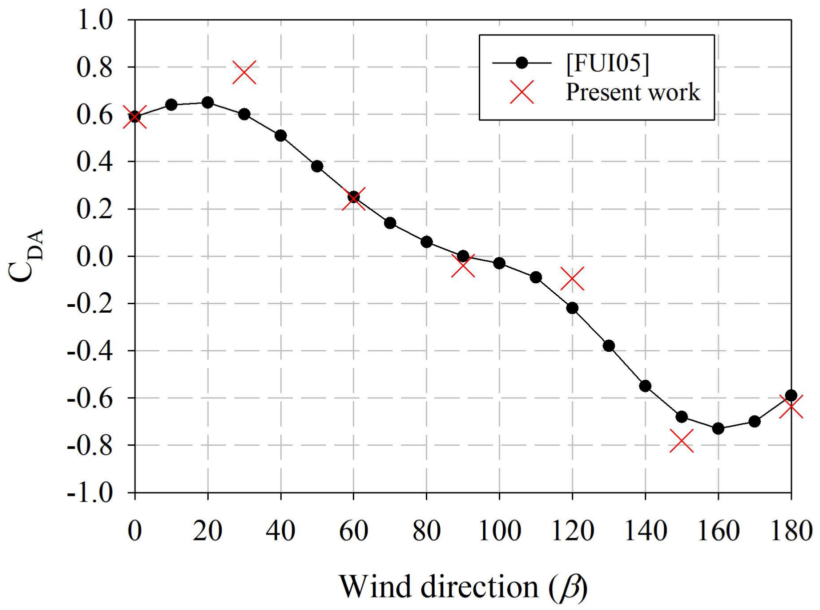


Fig. 2.44 Comparison of the air resistance using the current scheme and the regression line proposed at [FUI05].

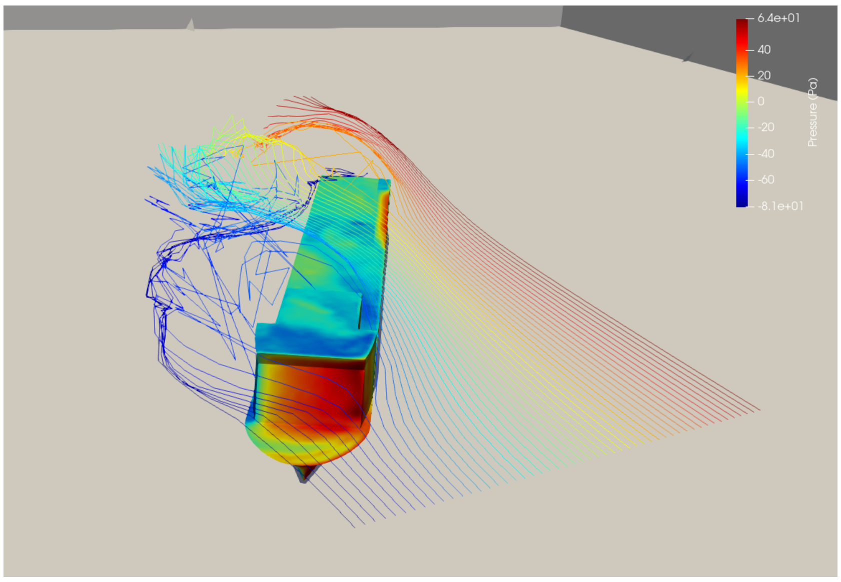
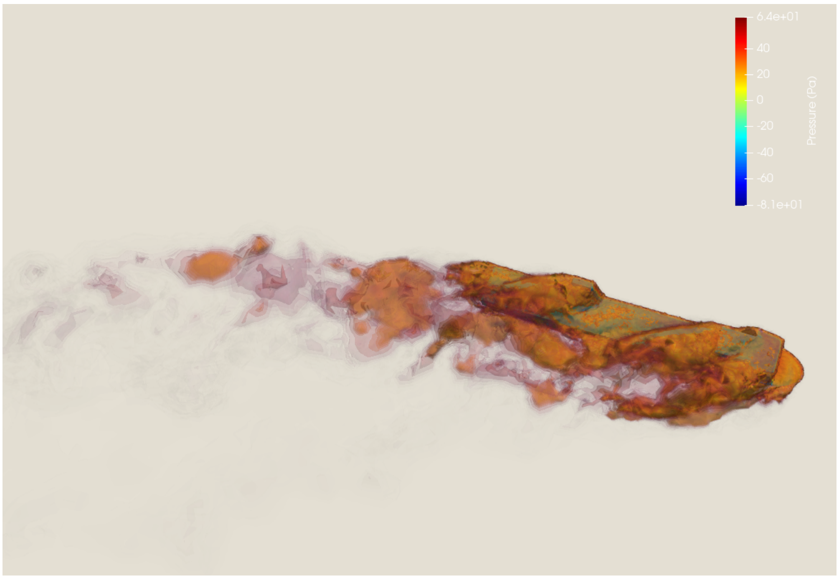
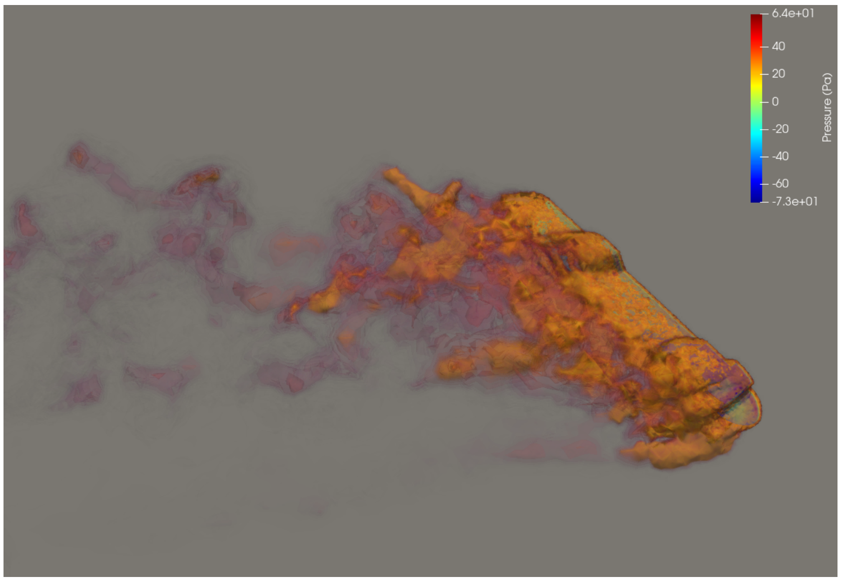


Fig. 2.45 Streak lines obtained for a 30 degree heading angle.



(a) 30 degree heading angle



(b) 60 degree heading angle

Fig. 2.46 Vorticity contours for 30 and 60 degree heading angle.

Chapter 3

Free Surface Problems

3.1 Introduction to interface problems

Up to now, a method to solve the incompressible navier-stokes problem for one fluid has been presented. But, this method, SL-PFEM, can expand its capabilities by solving problems with interfaces. These types of problems can be of:

1. Multifluid or free surface problems.
2. Fluid Solid interaction.
3. Free Surface and Solid interaction.

When an interface problem is present in a fluid simulation, a discontinuity can appear at the pressure gradients such as the free surface problem or at the velocities such as a fluid-solid problem. These discontinuities must be correctly computed to prevent numerical errors. In fact, [Her09] studies this error when applied to free surface problems where an incorrect capture of the pressure gradient jump due to the free surface provokes a degradation to the solution that loses the mass conservation property.

So the main objective with these types of problems is to correctly capture the position of the interface and correctly apply the corresponding boundary conditions. An initial and known way of doing this would be to generate a set of points at the intersection of the interface with the background mesh and carry out a local mesh refinement so that it takes into account the boundary conditions for each fluid phase. This method can prove to be quite costly since adding new intersection points in order to catch the fluid-fluid interface forces the number of degrees of freedom to grow. This increase means that the discrete system of the equations must increase their size to take into account the new degrees of freedom. This change of size increases the computational effort as assembling the different matrices representing the discrete system of equations is very expensive. In fact, this will be done each time the interface changes meaning that for moving interfaces, such as the fluid-fluid interfaces, the system matrices will have to be reassembled at each time-step.

This task of continuously changing the system matrices is what makes this initial method so computationally expensive to implement. So, an alternative to the refinement method is to use enrichment [Her09], where new degrees of freedom are introduced with no need of increasing the size of the discrete system of equations to be solved. The enrichment method developed within this thesis will be capable of coping with fluid-fluid and solid-fluid interfaces simultaneously.

3.2 Enrichment

The objective when using the enrichment method is to prevent the resizing of system matrices. To do this, an approximation to the mesh refinement methodology is used.

When an interface problem is present, elements are intersected as follows:

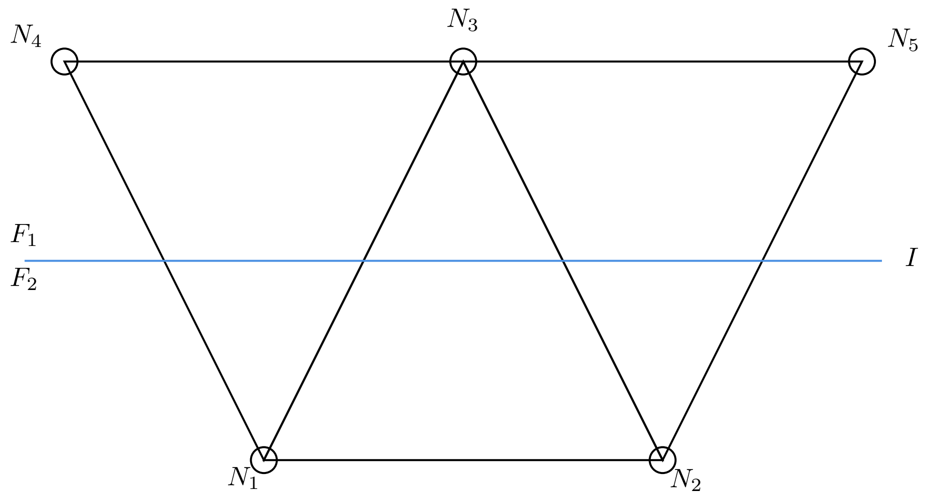


Fig. 3.1 Free surface interface (blue) over the background mesh (black)

The first step when using either the refinement or the enrichment methods is to find the intersections of the interface with the mesh (see figure 3.2) where the intersections are defined by I_i being $i = 1, 2, 3, \ldots$. The next step is to unassemble the elements that are intersected by the interface. Finally, the last step that is equal to the two methods is creating the sub-elements (figure 3.3) where a series of auxiliary nodes are generated at the centre of the non-intersected edge marked as a_i.

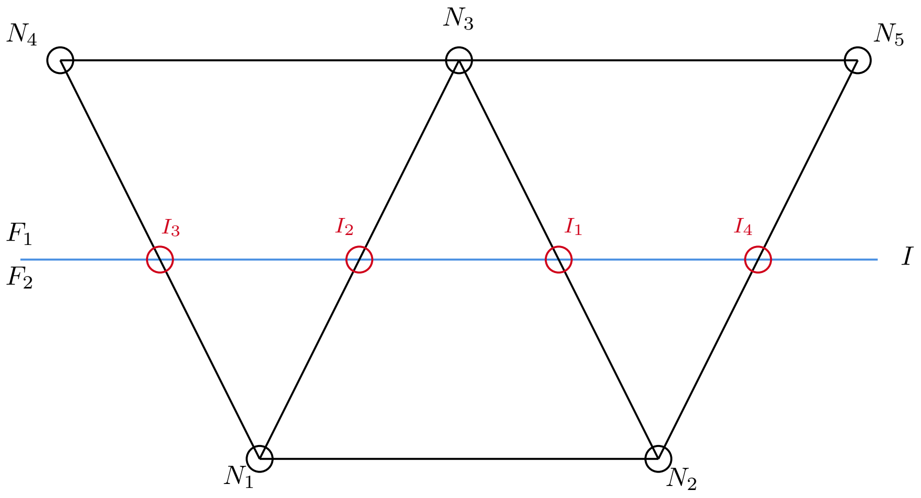


Fig. 3.2 Intersections obtained of an interface with the background FE mesh.

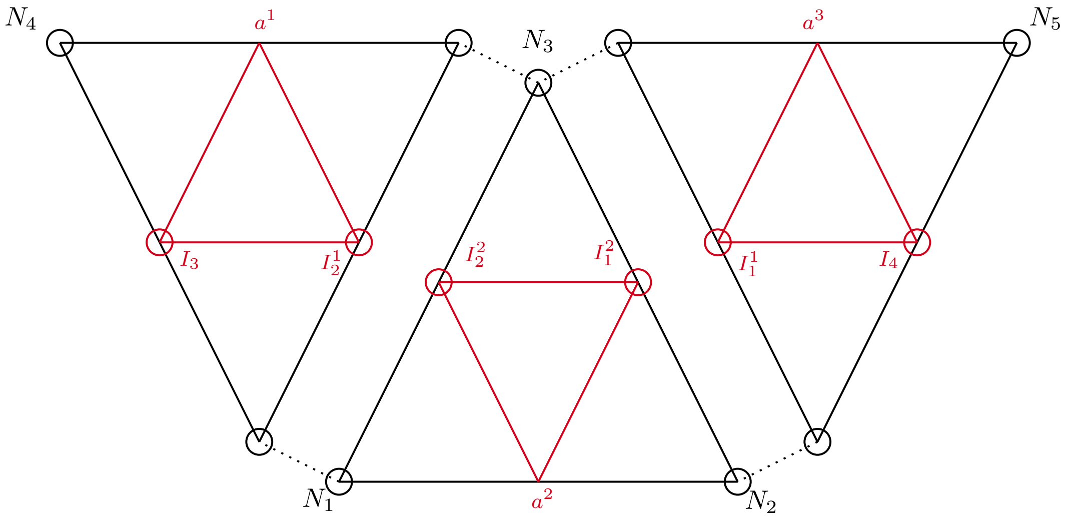


Fig. 3.3 Sub-Elements obtained when using an enrichment or refinement method.

Once the sub-elements have been obtained, the two methods diverge. On one hand, the refinement method will directly assemble the new degrees of freedom (intersections). To better see this, a system of equation is presented corresponding to the refinement of the zone where the interface is at can be written as:

$$\begin{bmatrix} \bar{\mathbf{L}}_{nn} & \bar{\mathbf{L}}_{nI} \\ \bar{\mathbf{L}}_{In} & \bar{\mathbf{L}}_{II} \end{bmatrix} \begin{bmatrix} \phi_n \\ \phi_I \end{bmatrix} = \begin{bmatrix} b_n \\ b_I \end{bmatrix} \tag{3.1}$$

Being n the set of original degrees of freedom and I the number of new degrees of freedom (intersections). In this case, the system of equations grows by the number of intersections meaning that the original system changes its shape and size. This means having to reassemble the system each time the interface changes which turns out to be computationally expensive.

On the other side of the bifurcation, we have the enrichment method which is used in this thesis. The general steps to follow when enriching the system are:

1. Find the intersections of the interface with the mesh.
2. Unassemble the original element from the main mesh.
3. Divide the element into the local sub-elements.
4. Construct the local system of equations.
5. Collapse the local system into the global one.

Being the first three steps the common steps with the refinement method and the last two the main difference with said method. As this method is used in the thesis, each step will now be further detailed in order to better understand the process of enriching a system of equations when an interface is present.

Find the intersections of the interface with the mesh

To begin the enrichment process, the first part is to find the interface and how it intersects with the mesh(figure 3.2). In this thesis two methodologies are needed depending on the type of interface:

- To compute the free-surface interface, a Semi-Lagrangian Particle Level-Set Method is developed and explained in detail later on in sub-section 3.3. Basically, the particles transport a level-set function that is projected onto the background mesh in a similar fashion as done with the particle's velocities.
- When computing fluid-solid interfaces, which is known at all times, the interface will be computed by performing mesh to mesh intersections. This is a precise method to find the interface and prevent errors due to an incorrect estimation of the solid boundary.

Unassemble the original element from the main mesh

This is done by performing the inverse assembly operation.

Divide the element into the local sub-elements

The idea is to compute the same equations on the new sub-elements using FEM. In fact, different ways to perform this are available, one of them being presented at [Her09] where for 2D cases the triangular elements are divided into three new triangles (figure 3.4). This discretisation allows to properly catch and compute an interface problem with a discontinuous function, for example the gradient discontinuity of a free surface problem.

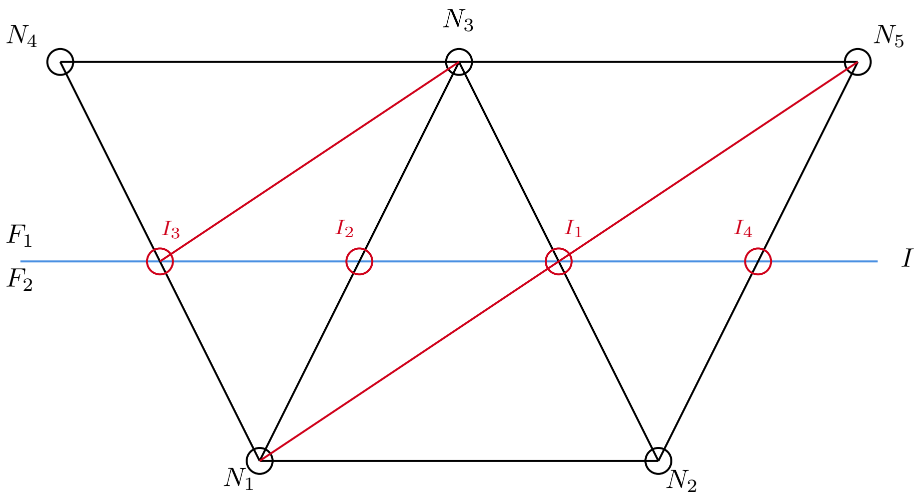


Fig. 3.4 Sub-elements proposed by [Her09]

The main disadvantage with this discretisation relies simply on the fact that it only allows for two intersection points, whereas cases such as solid-free surface interactions intersections on all of the edges of a 2D triangle can be found, see figure 3.5. In these cases more sub-elements are needed in order to take into account the two interfaces. Figure 3.6 presents the sub-elements used to obtain the interfaces of figure 3.5 and that a final four sub-elements are constructed in 2D cases (a total of eight sub-elements in 3D cases). Therefore, figure 3.3 shows the general construction of the sub-elements. If only one interface is present, auxiliary degrees of freedom are inserted at the mid points of the edges where no intersection is present.

Remark: When one intersection is very close to a node, then one of the triangles has an area close to zero with a bad aspect ratio causing the system matrix to be ill-conditioned. Therefore, said triangle is removed collapsing the intersection with the

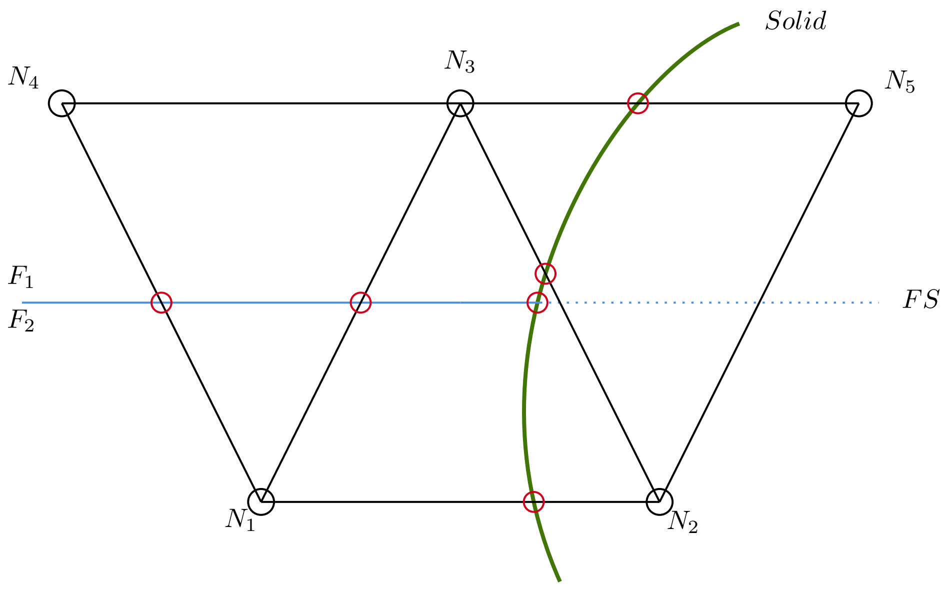


Fig. 3.5 Two interfaces intersecting all of the edges of one element.

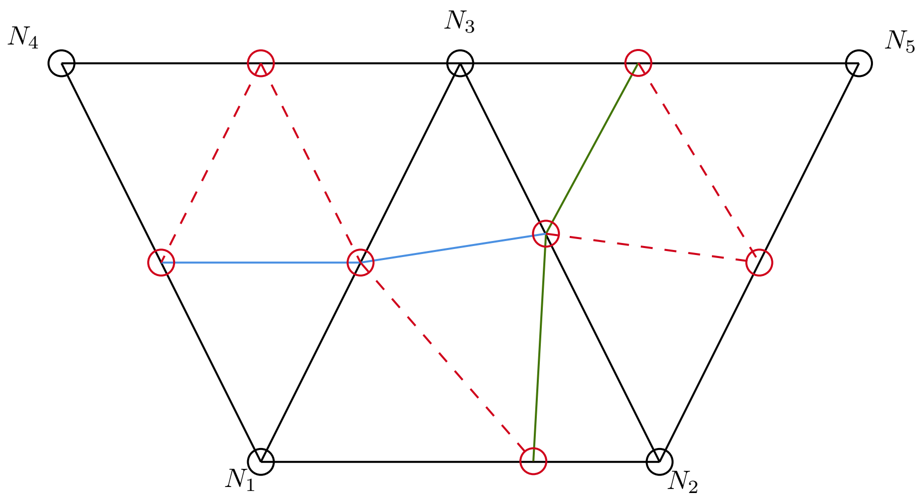


Fig. 3.6 Construction of sub-elements in the case of two interfaces intersecting all of the edges of one element.

nearest node into one single degree of freedom. Figure 3.7 shows this problem and the procedure to circumvent it.

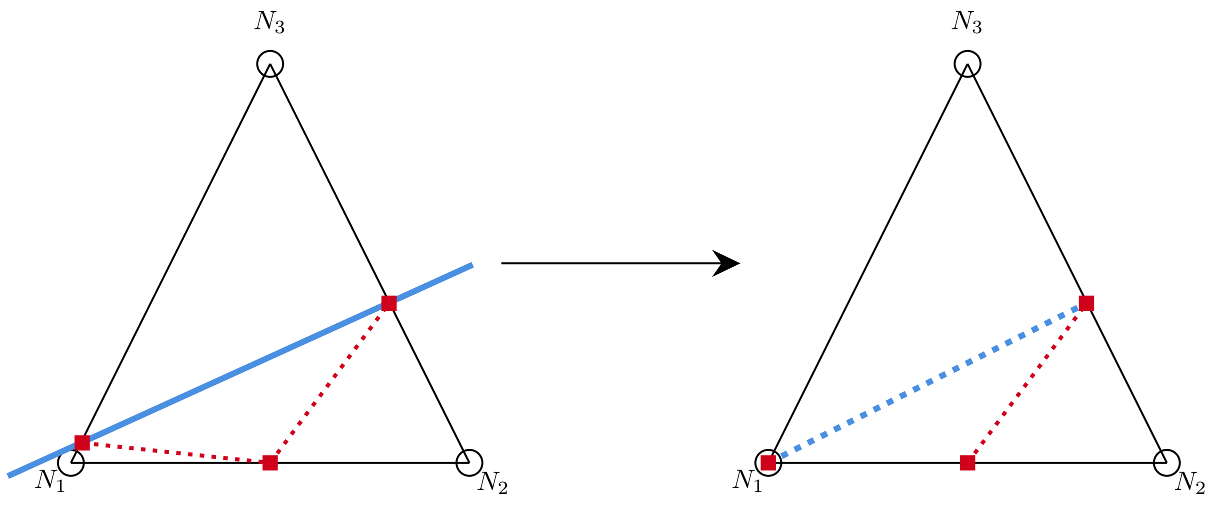


Fig. 3.7 Small local sub-element is removed.

Construct the local system of equations

Once the element has been divided into the sub-elements, the equations can be built locally by assembling these new elements into a local system of equations. As a general rule, variables that belong to the new degrees of freedom will be named the same as the main variables but with the subscript $\star$ and matrices will be will be divided taking into account this same nomenclature.

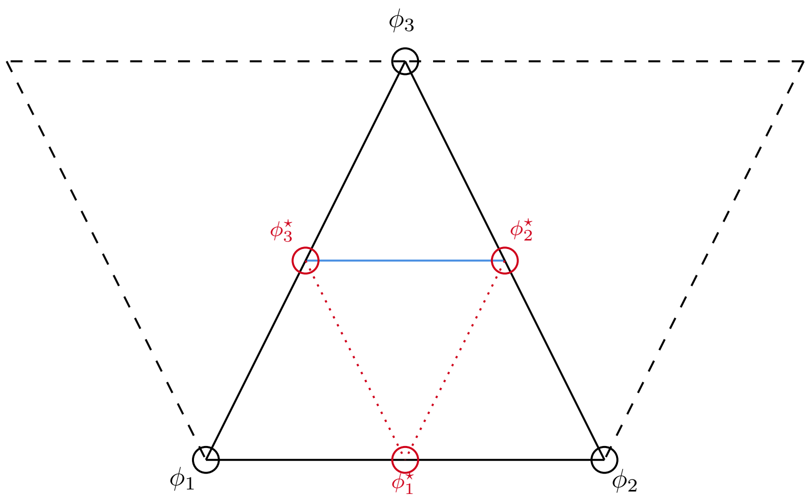


Fig. 3.8 Nomenclature used for each of the nodes.

With the nomenclature shown at figure 3.7 a system of equations for a Poisson problem, $\Delta\,\phi = f$, discretised using FEM can be presented for the local enriched element as:

$$\begin{gathered}\begin{bmatrix}\bar{\mathbf{L}}_{kk} & \bar{\mathbf{L}}_{k\star}\\ \bar{\mathbf{L}}_{\star k} & \bar{\mathbf{L}}_{\star\star}\end{bmatrix}\cdot\begin{bmatrix}\phi_k\\ \phi_\star\end{bmatrix}=\begin{bmatrix}f_k\\ f_\star\end{bmatrix}\\ k:\phi_1,\,\phi_2,\,\phi_3\\ \star:\phi_1^\star,\,\phi_2^\star,\,\phi_3^\star\end{gathered}\tag{3.2}$$

Where it can be seen how the Laplacian matrix for the enriched element is divided into parts to take into account the elements created using the new enriched degrees of freedom.

Collapse the local system into the global one

This step is the last one and collapses the new degrees of freedom. Collapsing the degrees of freedom consists of obtaining an explicit dependency of the new degrees on the old ones. This prevents having to grow the system matrices by the new degrees of freedom.

To show how this collapse is done, the Poisson equation presented above is used. The idea behind the collapse is to locally insert the new degrees of freedom:

$$\phi_\star = \bar{\mathbf{L}}_{\star\star}^{-1}\cdot\left(f_\star - \bar{\mathbf{L}}_{\star k}\cdot\phi_k\right)\tag{3.3a}$$

$$\bar{\mathbf{L}}_{kk}\cdot\phi_k = f_k - \bar{\mathbf{L}}_{k\star}\cdot\phi_\star\tag{3.3b}$$

And inserting 3.3a into the equation of 3.3b:

$$\left[\bar{\mathbf{L}}_{kk} - \bar{\mathbf{L}}_{k\star}\cdot\bar{\mathbf{L}}_{\star\star}^{-1}\cdot\bar{\mathbf{L}}_{\star k}\right]\cdot\phi_k = f_k - \bar{\mathbf{L}}_{k\star}\cdot\bar{\mathbf{L}}_{\star\star}^{-1}\cdot f_\star\tag{3.4}$$

It is worth mentioning that the local matrices such as $\bar{\mathbf{L}}_{\star\star}$ are small enough that a direct inverse of them is feasible from a compute perspective. In a 2D case, the matrix size is $3x3$ and in 3D $7x7$.

As a final remark, figure 3.3 shows each element with its constructed sub-elements. By performing the local collapse a duplicity appears at the intersections. The problem relies on an intersection that is shared by two neighbouring elements. This intersection is collapsed locally at each element and means that the obtained result can differ from one element to another.
When applying Dirichlet boundary Conditions at the interface, this problems disappears

and in fact makes the enriched method to be the same as the refinement method. But, when applying Neumann boundary conditions, then the boundary integrals are taken into account in order to minimise the error produced at the duplicated intersections. This last solution is further analysed for each interface problem type (fluid-fluid or solid-fluid).

3.3 Semi-Lagrangian Particle Level-Set

This section presents the methodology used to capture the free surface interface. To do this an implementation of the Level-Set method is applied to the Semi-Lagrangian scheme and will be called Semi-Lagrangian Particle Level-Set Method (SL-PLSM). In this case, a level-set function is used to represent the interface that has to be captured.

This level-set function is represented over the particles and is defined using signed functions, see figure 3.9. Such functions have the particularity that at the interface the value of the function is 0 and the fluid at each side of it have an opposite signs:

$$\begin{aligned} f\left(x\right)>0 \qquad & if \qquad x\in\Omega^{+} \\ f\left(x\right)<0 \qquad & if \qquad x\in\Omega^{-} \\ f\left(x\right)=0 \qquad & if \qquad x\in\Gamma \end{aligned} \tag{3.5}$$

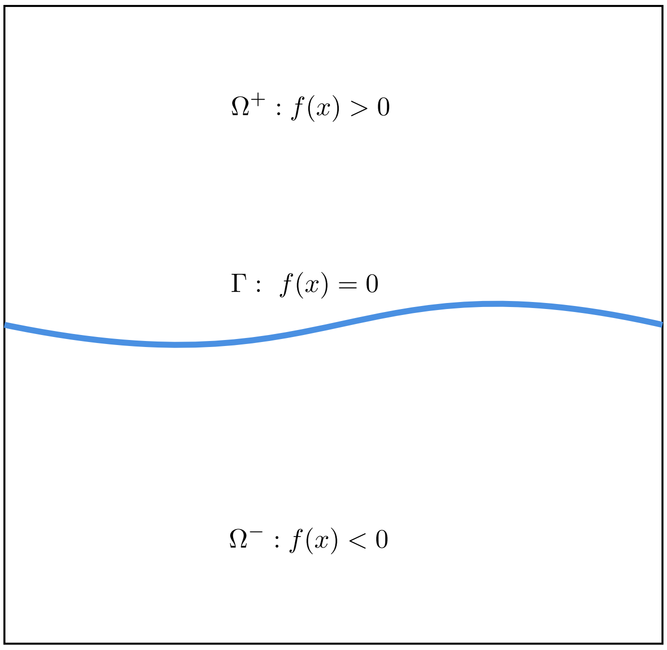


Fig. 3.9 Signed function used for the SemiLagrangian Particle Level-Set.

3.3.1 Level-Set functions and advantages

A variety of functions can represent the signed function and, in this thesis, two of them are used: A step function (equation 3.6) and a smooth vertical distance function (equation 3.7). While both types of function can define an accurate interface over the particle set, the same cannot be said when the functions carried by the particles are projected to the FE mesh.

The first signed function, the step function (3.6), is very easily defined as one side of the interface has a value of 1 and the other a value of -1, allowing to define the interface at value 0. The ability to define with ease the function allows to easily re-seed particles as the value of the level-set is easily obtained.

$$\begin{aligned} f(x) &= 1 \qquad if \qquad x \in \Omega^+ \\ f(x) &= -1 \qquad if \qquad x \in \Omega^- \end{aligned} \tag{3.6}$$

The second signed function, the smooth distance function (3.7), is also simple to define as in this case it represents the vertical distance of the particle with respect to the position of the interface. The advantage of such a function is it's smoothness and can therefore capture the interface with a high degree of accuracy.

$$f(x) = y - \eta(x) \tag{3.7}$$

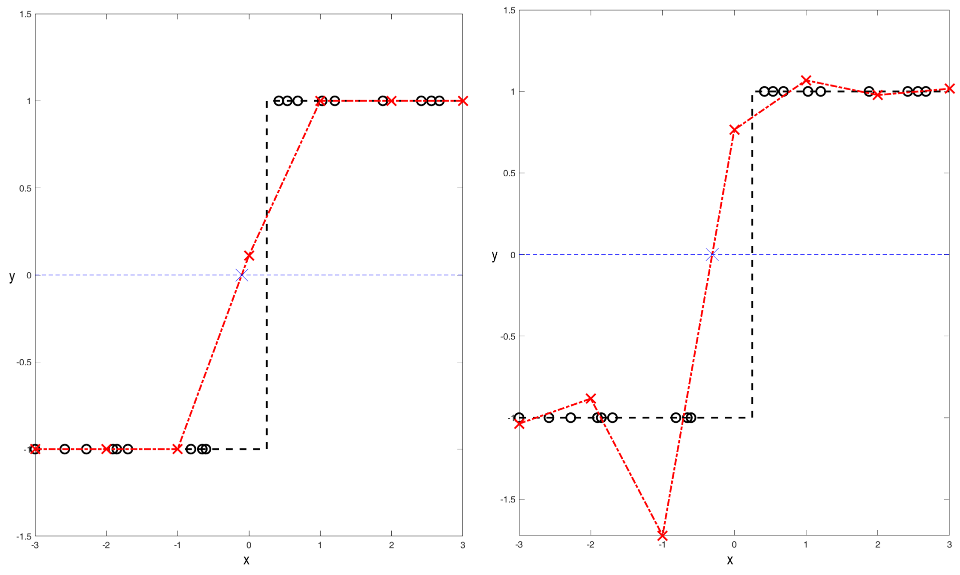
η represents a smooth free surface wave type propagating in the OX direction. The two functions are easy to initialise once the interface position is known. In fact, the step function can be started by defining the area (2D) or volume (3D) of one fluid and the other without having to define the interface position. On the other hand, the smooth distance function needs to have an exact solution of the initial interface position which proves to be difficult in some cases.

3.3.2 Level-Set functions disadvantages

This subsection analyses the performance of the two functions used for the SL-PLSM based on two main sources of errors: Errors appearing from projecting the functions from the particles to the FE Mesh and a mixture effect of the functions produced over the particles.

3.3.2.1 Level-Set functions: Projection errors

The step function proves to be difficult to capture the interface as the projection of the step function commits an error that can be as large as the distance between two particles. A 1D case is used to presents this problem where the interface point crossing $y = 0$ is found at $x = 0.25$ discretised using a total of 6 elements. For the step function, three



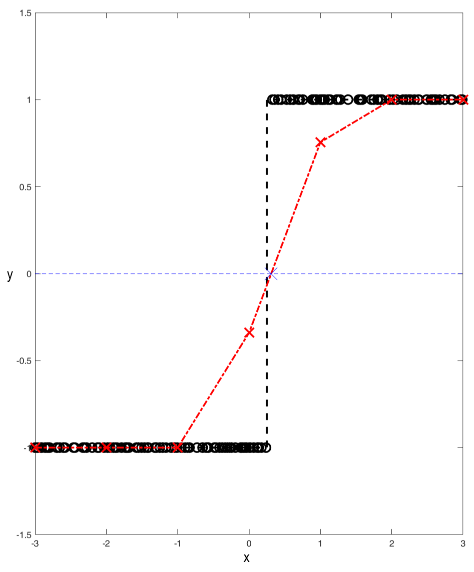
(a) RMSE $x_a - x_p = 0.3510$ (b) RMSE $x_a - x_p = 0.5576$

Fig. 3.10 Step function projection using 3 particles per element. To the left Node Local Least-Squares, and right the Global Least-Squares

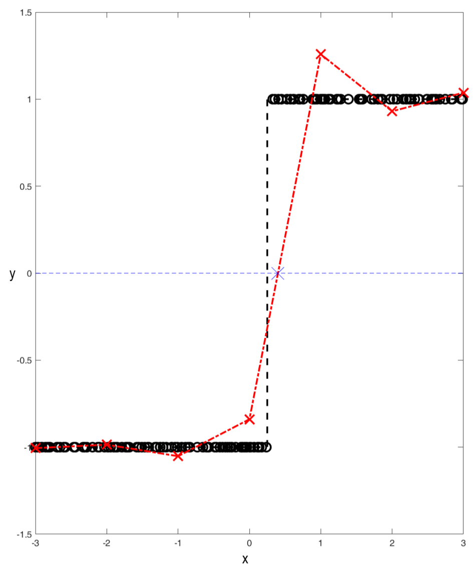
different cases are presented where the difference relies on the amount of particles per element used. All three use the Node Local Least-Squares and the Global Least-Squares for projecting the particle value to the mesh.

Figures 3.10, 3.11 and 3.12 present the results obtained using 3, 30 and 300 particles per element respectively. The black dashed line represents the result that the particles (black circles) transport while the red line formed by the red crosses present the projected results over the 1D mesh. They all show how the error between the expected position of the interface x_a and the obtained one x_p slowly decreases by incrementing the amount of particles.

The same analysis is performed using a signed smooth distance function, in this case the distance between the particle's position and the expected position of the interface. The previous chapter showed how a smooth function is recovered exactly, and in a 1D problem where the smooth function is linear then the obtained projected result is exact with 0 error.

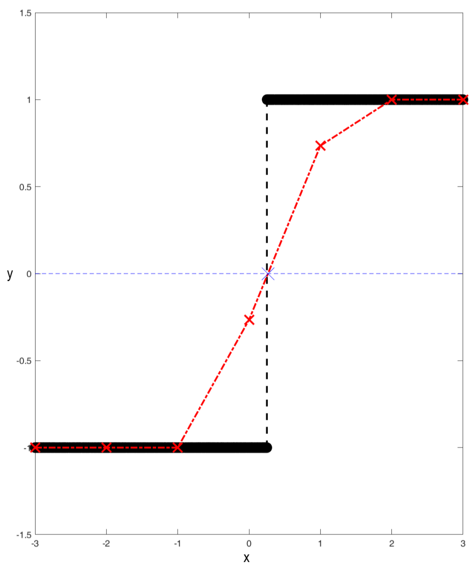


(a) RMSE $x_a - x_p = 0.0603$

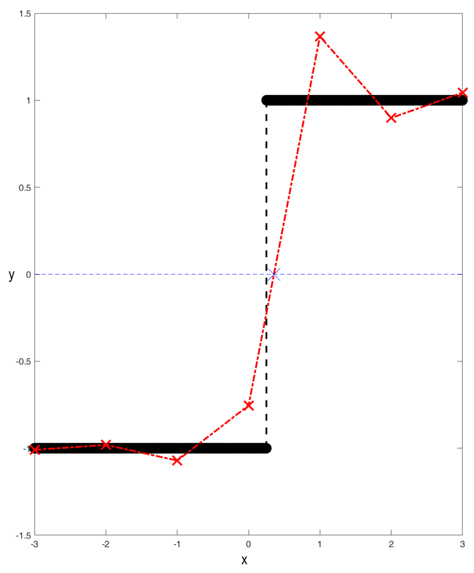


(b) RMSE $x_a - x_p = 0.1502$

Fig. 3.11 Step function projection using 30 particles per element. To the left Node Local Least-Squares, and right the Global Least-Squares



(a) RMSE $x_a - x_p = 0.0148$



(b) RMSE $x_a - x_p = 0.1056$

Fig. 3.12 Step function projection using 300 particles per element. To the left Node Local Least-Squares, and right the Global Least-Squares

3.3.2.2 Level-Set functions: Mixture effect

Another problem that is common for the two presented level-set functions appears when particles with different function values mix up. When projecting the step function, an error is committed which produces an inconsistency of the level-set function over the particles and the one projected over the mesh. This means that particles of one fluid can find themselves at the other side of the interface causing a mixture effect.

But when projecting the smooth distance function, this mixture is not as evident as the step function as the obtained interface is extremely accurate. In fact, the mixture appears when particles move within their fluid type. For the step function this would not be a problem as particles moving within the same fluid have a constant value, but in the case of a smooth distance function this changes a lot. One particle with a very large level-set value can find itself very close to the interface producing projection errors.

This mixture has a negative effect, especially due to particle re-seeding. If the interface is badly captured over the FE mesh then re-seeded particles will be incorrectly initialised as their initial value is interpolated from the mesh (figure 3.13).

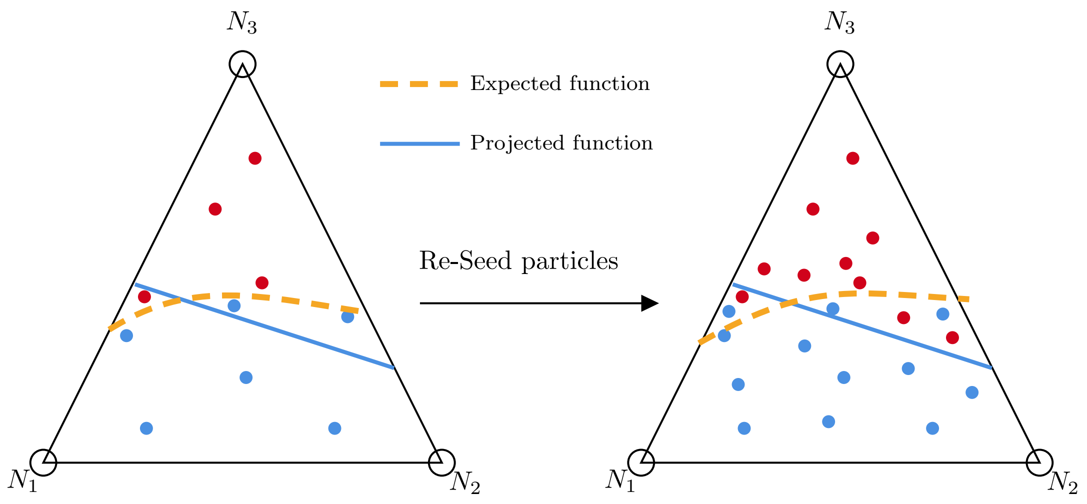


Fig. 3.13 Badly initialised Re-seeded particles due to the projection error.

3.3.3 Interface capturing: Error correction

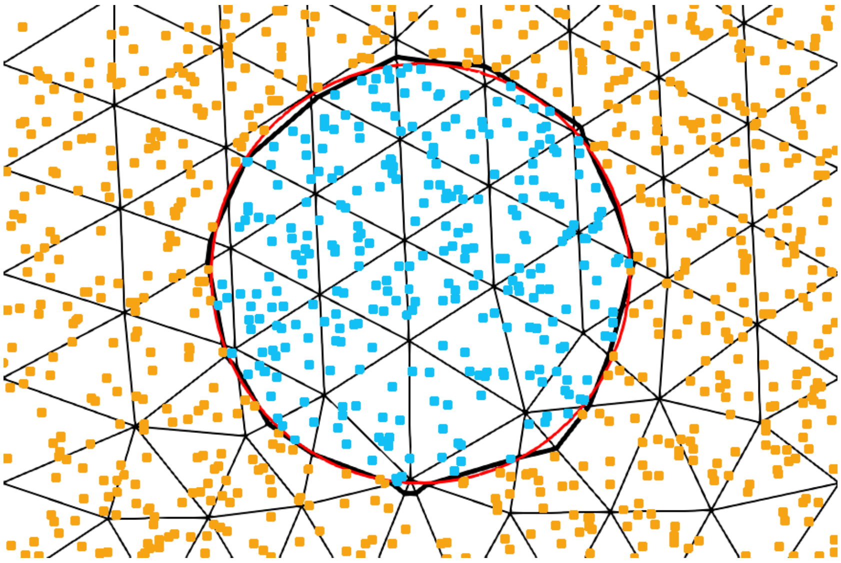
This subsection tries to reduce the projection error when using the step function. Using the step function allows to very easily define a domain compared to the signed smooth distance function. Therefore, in order to improve the step function projection different stages are performed.

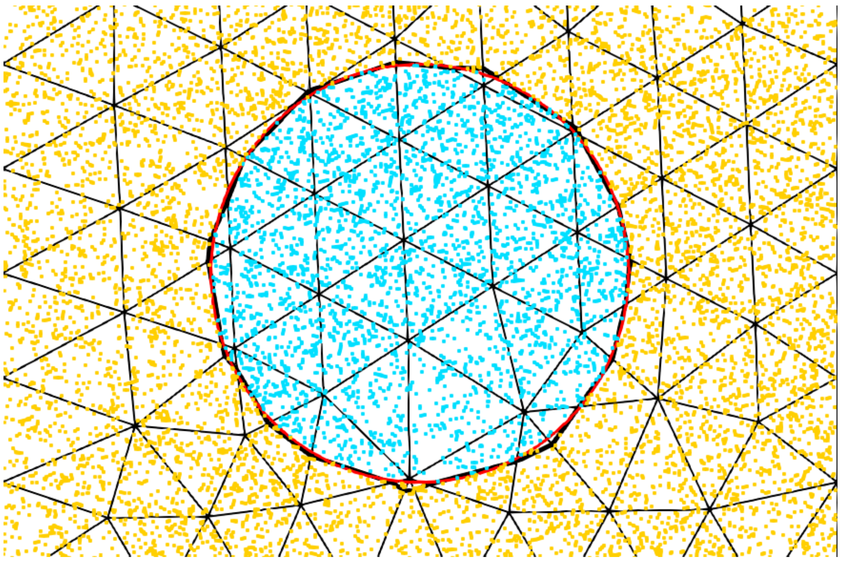
First, the step function is projected using the Node-Local Least-Squares projector. This projector, though inserting diffusion at the step, serves as an initial estimation of the interface. When projecting the step function with a simple projection operation, then the error depends on the number of particles used to approximate the interface. Figure 3.14 shows how the interface is more accurate when increasing the number of particles per element.

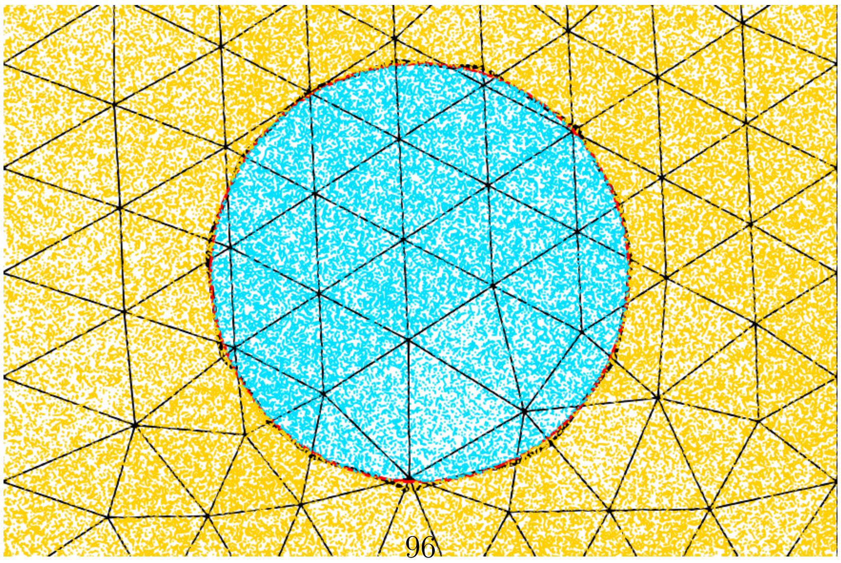
The second stage consists in performing a two step iteration scheme that tries to correct the projection error. The two steps are:

- Level-Set function interpolation. This step takes the value of the level-set function on the mesh and interpolates it back to the particles, therefore performing an update of the particle level-set function. Only particles that belong to the same side of the projected function are updated, the ones that find themselves with a different sign than the interpolated function are not updated. For example, figure 3.15a showcases this, where the projected level-set function (black) has a large error compared to the initial interface. Therefore, the particles found within the red circle have a different sign of the function compared to the interpolated one. In this case, these particles will not update the distance variable they transport, while the rest of the particles will have this variable updated.
- Level-Set function projection. Once the level-set function has been interpolated back to the particles, a new projection using the Global Least-Squares projector is used.

Figure 3.15 shows 3 snapshots of the iteration scheme. As it can be seen, the captured interface improves drastically after 5 iterations. If no iterations where performed (figure 3.15a) then a decrease in area of the circle would be present which, for free surface problems, means that a mass loss would be present.

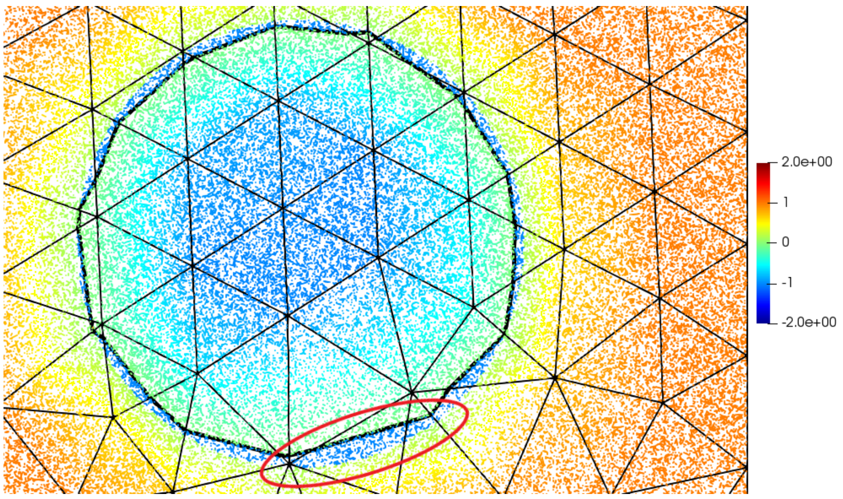

(a) Circle projection using 10 particles per element.


(b) Circle projection using 100 particles per element.

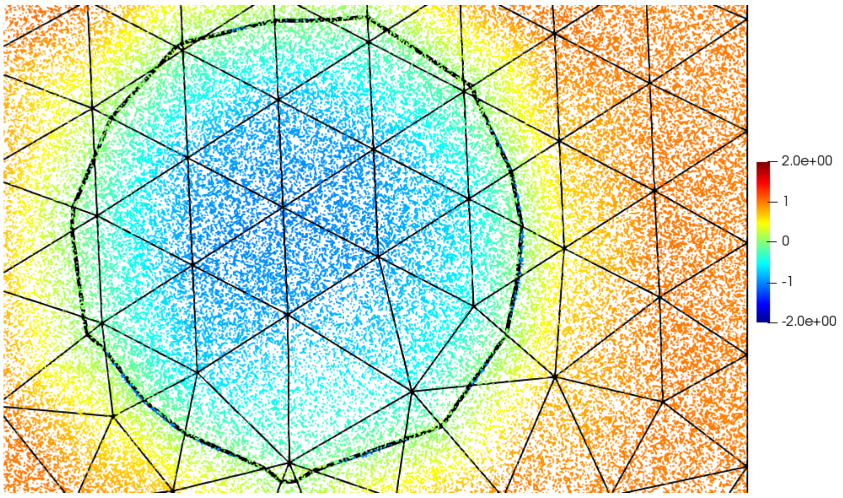


(c) Circle projection using 1000 particles per element.

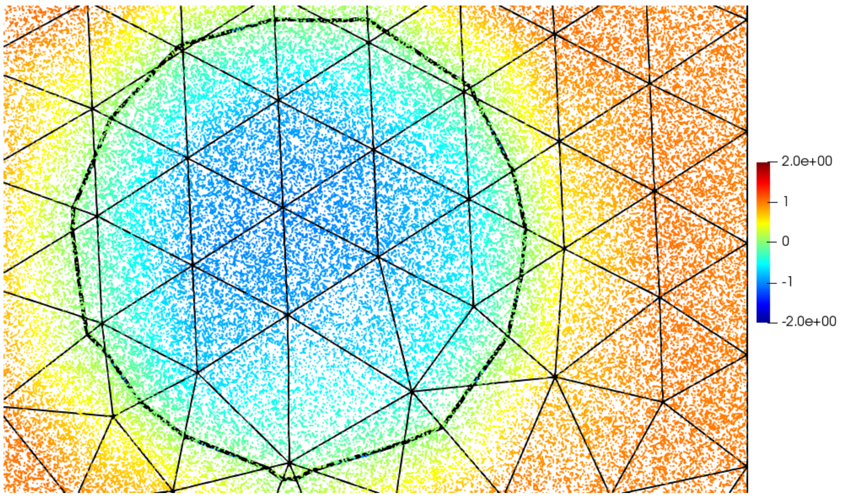
Fig. 3.14 Mesh used for the 2D flow around the cylinder.



(a) Error correction at the first iteration.



(b) Error correction after 2 iterations.



(c) Error correction after 5 iterations.

Fig. 3.15 Variation of the circular interface at different iterations of the error correction scheme.

3.3.4 Reinitalisation of the Level-Set Functions

Due to the mixture and badly initialised re-seeded particles, noise appears at the interface that degrades the final solution. To prevent this, the level-set function of the particles must be re-initalised, for example, by interpolating at each time-step the projected level-set function of the mesh over the particles.

This re-initalisation method also has a negative effect as it inserts diffusion into the level-set function transported by the particles when using projectors that do not comply with the coherence condition. When using the Global Least-Squares, this condition is fulfilled, therefore diffusion due to the projection-interpolation scheme is not a problem. But the smooth distance function slowly degrades due to the mixture effect, obtaining gradients of the function that tend to 0 which prevent obtaining an accurate solution of the interface.

Therefore, a different re-initalisation scheme has to be used that tries to accurately capture the interface over the mesh, reducing the diffusive effect of the interpolation and obtaining gradients that are not close to 0.

3.3.5 Reinitialisation by reconstructing the level set function

As stated before, due to a mixture effect the gradients close to the interface can tend to 0 even when performing a complete interpolation of the level-set function over the particles. This means that noise can appear at the interface when being captured using the scheme proposed at 3.3.3.

Ideally, the good approach would be to obtain a smooth distance level-set function that truly behaves as a distance function (not a vertical distance to the interface position). For example, a circular distance function would be seen as shown at figure 3.16.

To do this, equation 3.8 is used which is discretised using FEM at equation 3.9 and is solved over the whole fluid domain which manages to obtain a good approximation of the smooth distance function. A Conjugate Gradient solver is used and at each iteration i the gradients of the level-set function are normalised. Therefore, one iteration is considered to be one iteration of the solver and the normalised gradients.

To show the performance of this method, a square domain is defined that contains a circular interface constructed by initially assigning a step-function to the nodes. By performing the aforementioned iterations the initial step function is smoothed without needing to add further boundary conditions.

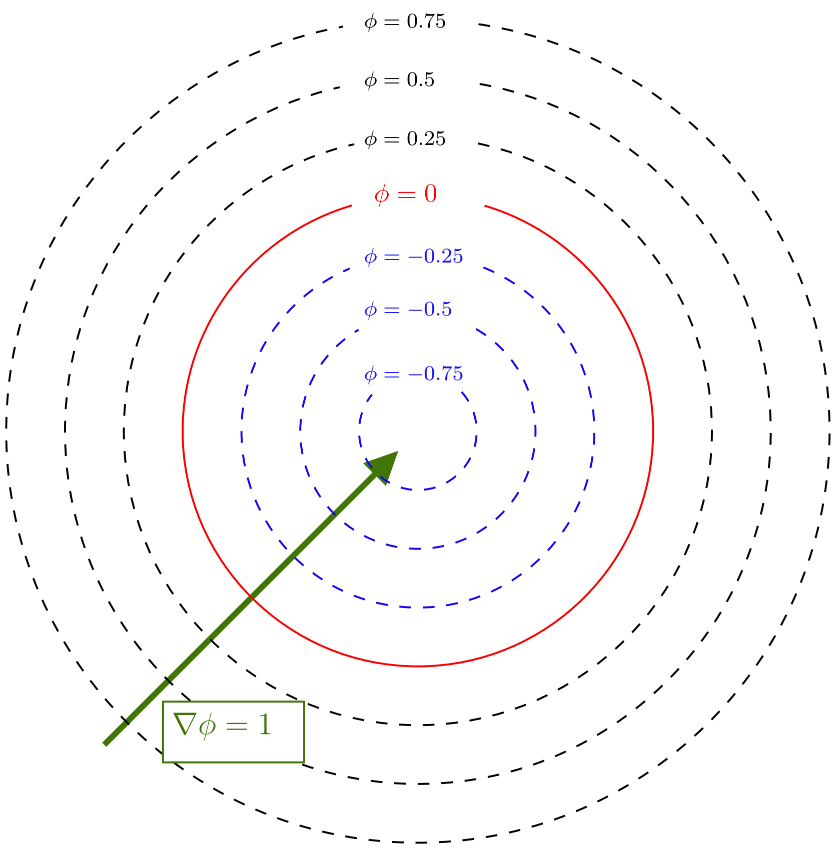
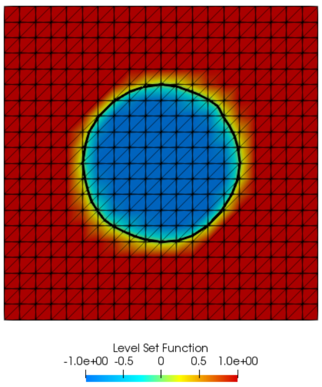


Fig. 3.16 Ideal Smooth distance Level-Set function for a circular interface.

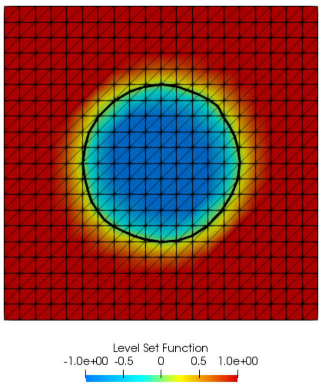
The figures shown at 3.17 show how with a different amount of iterations the projected values of the distance function become smooth which is the objective in this section. In fact, figure 3.18 shows the contours and how after 1000 iterations of equation 3.8 a good smooth distance function for the circular interface is obtained.

$$\Delta d^{i+1} = \nabla \cdot \frac{\nabla d^i}{|\nabla d^i|} \tag{3.8}$$

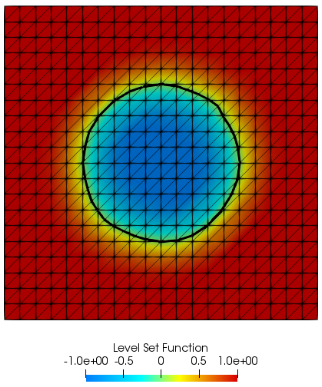
$$\bar{\mathbf{K}}\, d^{i+1} = \bar{\mathbf{D}}^{\,T} \frac{\bar{\mathbf{G}}\, d^i}{\left|\bar{\mathbf{G}}\, d^i\right|} \tag{3.9}$$



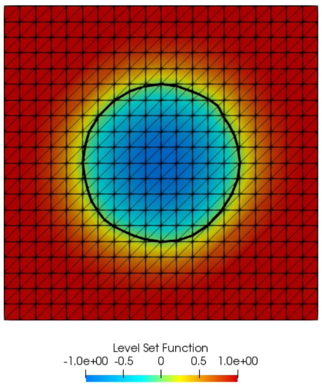
(a) Solver iterations: 0.



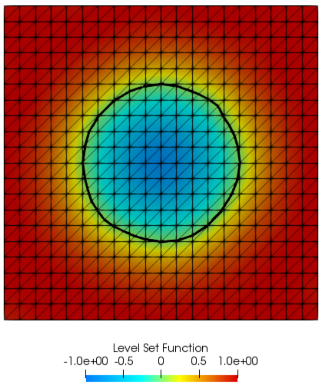
(b) Solver iterations: 1.



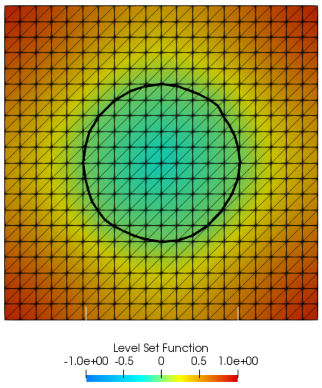
(c) Solver iterations: 2.



(d) Solver iterations: 5.



(e) Solver iterations: 10.



(f) Solver iterations: 100.

Fig. 3.17 Variation of the interface after a certain amount of solver iterations.

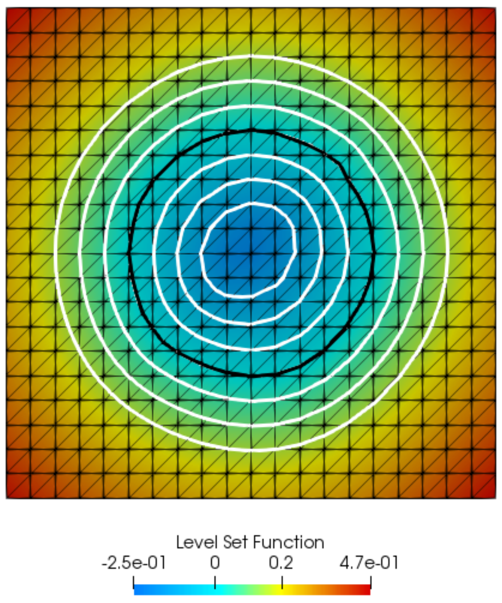


Fig. 3.18 Circular interface: contours after 1000 iterations.

3.4 Free Surface problems

When solving a free surface problem, a discontinuity at the pressure appears forcing a jump at the pressure gradients due to the density difference. But if the pressure gradients are computed as the gradient of the pressure over the density, then a continuous function for the normal pressure gradient can be computed. This way, the discontinuity only appears at the pressure equation of the Navier-Stokes equations. The normal stresses that affect the free surface are the ones coming from the normal pressure gradient and the normal viscous term. Therefore, at the interface we assume continuity of these stresses, of the pressure variable and the normal velocity:

$$\begin{gathered} -\frac{\nabla P_1}{\rho_1}\mathbf{n} + \nu_1 \Delta \mathbf{u}_1 \cdot \mathbf{n} = -\frac{\nabla P_2}{\rho_2}\mathbf{n} + \nu_2 \Delta \mathbf{u}_2 \cdot \mathbf{n} \\ P_1 = P_2 \\ \mathbf{u}_1 = \mathbf{u}_2 \end{gathered} \tag{3.10}$$

In fact, for this particular thesis where the objective is to reach a point where seakeeping hydrodynamics can be computed, water and air will be the two fluids involved. These two fluids have a density difference of about one thousand times, and we can assume that the viscous effects at the free-surface and the free-surface tension are negligible. Therefore the above equations can be written as:

$$\frac{\nabla P_1}{\rho_1}\mathbf{n} = \frac{\nabla P_2}{\rho_2}\mathbf{n} \tag{3.11a}$$

$$\mathbf{u}_1 \cdot \mathbf{n} = \mathbf{u}_2 \cdot \mathbf{n} \tag{3.11b}$$

$$P_1 = P_2 \tag{3.11c}$$

Meaning that at the interface, the normal pressure gradient must be continuous and a slip boundary condition is needed for the velocity. Also, as a remark, neglecting the viscosity effects at the free surface allows the air phase to slip over the water phase, creating a velocity discontinuity (Slip Boundary Condition). This will be further explained in section 3.5

To implement the enrichment method, first we must recall the pressure equation to be solved for one fluid:

$$\bar{\mathbf{K}} \left(\frac{P^{n+1,i+1}}{\rho} \right) = -\frac{2}{\Delta t} \bar{\mathbf{D}}\, \hat{\mathbf{u}}^{n+1,i+1} + \bar{\mathbf{D}}^{T} \bar{\mathbf{M}}_{L}^{-1} \bar{\mathbf{G}} \left(\frac{P^{n+1,i}}{\rho} \right) \tag{3.12}$$

Then for two fluids, the equation can be re-written by separating the two fluids as:

$$\begin{aligned}\bar{\mathbf{K}}\left(\frac{P_1^{i+1}}{\rho_1}\right)+\bar{\mathbf{K}}\left(\frac{P_2^{i+1}}{\rho_2}\right)=&-\frac{2}{\Delta t}\left(\bar{\mathbf{D}}\,\hat{\mathbf{u}}_1^{n+1,i+1}+\bar{\mathbf{D}}\,\hat{\mathbf{u}}_2^{n+1,i+1}\right)\\&+\bar{\mathbf{D}}^T\,\bar{\mathbf{M}}_L^{-1}\,\bar{\mathbf{G}}\left(\frac{P_1^{i+1}}{\rho_1}\right)+\bar{\mathbf{D}}^T\,\bar{\mathbf{M}}_L^{-1}\,\bar{\mathbf{G}}\left(\frac{P_2^{i+1}}{\rho_2}\right)\end{aligned}\tag{3.13}$$

As it can be seen, the above equations show how for each fluid, the pressure is divided by the fluid's density. In fact, to simplify the calculation process, the matrices $\bar{\mathbf{K}}$ and $\bar{\mathbf{G}}$ are divided by the nodal density:

$$\frac{\bar{\mathbf{K}}}{\rho_f}P_f^{i+1}=-\frac{2}{\Delta t}\bar{\mathbf{D}}\,\hat{\mathbf{u}}_f^{n+1,i+1}+\bar{\mathbf{D}}^T\,\bar{\mathbf{M}}_L^{-1}\frac{\bar{\mathbf{G}}}{\rho_f}P_f^{i+1}\tag{3.14}$$

3.4.1 Free-surface enrichment process

To put together the two fluids, enrichment is used at the interface. To apply the enrichment method for free-surface problems, the method proposed at the beginning of the chapter is used. A slight difference appears when applying it here, which is when the systems have unassembled the original elements from the main mesh then the pressure and gradient systems are divided nodally by the density.

While most of the enrichment steps are applied exactly the same way as the generic methodology presented at 3.2, three of them need further explanation as some changes appear due to the particularities of the free-surface problems.

1. Find the intersections of the free-surface interface with the mesh.
2. Divide the element into the local sub-elements.
3. Unassemble the original element from the main mesh.
4. Divide $\bar{\mathbf{K}}$ and $\bar{\mathbf{G}}$ matrices by density.
5. Construct the local system of equations.
6. Collapse the local system into the global one.
7. Construct and collapse the enriched right hand side.

Find the intersections of the free-surface interface with the mesh

To compute the free-surface problem, a Semi-Lagrangian formulation of the level set method is used allowing to avoid using stabilisation techniques for the convection term and the corresponding diffusive effect of such techniques. This methodology is presented at the previous section.

Construct and collapse the enriched right hand side

This part is particular for the free-surface as it constructs the right hand side for the local problem. Recalling the enrichment method, the local assembled system of equations is:

$$\begin{bmatrix} \bar{\mathbf{K}}_{kk} & \bar{\mathbf{K}}_{k\star} \\ \bar{\mathbf{K}}_{\star k} & \bar{\mathbf{K}}_{\star\star} \end{bmatrix} \begin{bmatrix} P_k \\ P_\star \end{bmatrix} = \begin{bmatrix} b_k \\ b_\star \end{bmatrix} \tag{3.15}$$

After collapsing the system, the following system of equations remains:

$$\bar{\mathbf{K}}_{kk} \cdot P_k = b_k - \bar{\mathbf{K}}_{k\star} \cdot \bar{\mathbf{K}}_{\star\star}^{-1} \cdot \left(b_\star - \bar{\mathbf{K}}_{\star k} \cdot b_k\right) \tag{3.16}$$

Where:

$$\begin{aligned} b_k &= -\frac{2}{\Delta t} \bar{\mathbf{D}}\, \hat{\mathbf{u}}_f^{n+1,i+1} + \bar{\mathbf{D}}^{\,T} \bar{\mathbf{M}}_L^{-1} \frac{\bar{\mathbf{G}}}{\rho_f} P_f^{i+1} \\ b_\star &= -\frac{2}{\Delta t} \left[\bar{\mathbf{D}}_{\star k}\; \bar{\mathbf{D}}_{\star\star}\right] \begin{bmatrix} \hat{\mathbf{u}}_k^{n+1,i+1} \\ \Im\left(\hat{\mathbf{u}}_k^{n+1,i+1}\right) \end{bmatrix} + \left[\bar{\mathbf{D}}_{k\star}^T\; \bar{\mathbf{D}}_{\star\star}^T\right] \begin{bmatrix} {\bar{\mathbf{M}}_L^{-1} \tfrac{\bar{\mathbf{G}}}{\rho_f} P_f^{i+1}}_k \\ \Im\left({\bar{\mathbf{M}}_L^{-1} \tfrac{\bar{\mathbf{G}}}{\rho_f} P_f^{i+1}}_k \right) \end{bmatrix} \end{aligned} \tag{3.17}$$

As it can be observed, $b_\star$ appears in the assembled term for the right hand side, so the way it is assembled is very similar to the original right hand side but has minor differences. The difference is that a duplicity of the intersection nodes appear at adjacent elements (presented at 3.2). The way to solve this problem comes by using the following two approximations:

- First, the velocity variables have not been enriched, therefore computed values are not available at the enriched nodes. To overcome this, the normal components are interpolated between phases, while the tangential components are extrapolated from each fluid phase and the corresponding phase contribution is assembled.
- The pressure gradient term performs the same operation as the velocity variable, where normal components are interpolated between phases, the tangential components are extrapolated from each phase and then they are assembled according to the corresponding phase contribution. This is also done in order to force the contour integral to be cancelled out between to adjacent elements.

As a final remark, the enriched right hand side will be computed at each iteration while the local systems only need to be assembled at the beginning of the time-step.

3.5 Wave particulars

So far it is assumed that the normal velocity field is continuous at the free surface. When dealing with ocean water waves, the viscous effects are negligible as well as the boundary layer created at the free surface. Then it can be assumed that the air phase slips over the water phase, introducing a discontinuity in the velocity. This discontinuity has to be dealt with, otherwise an artificial viscosity effect will appear at the free surface.

Kinematic BC:
$$\frac{d\eta}{dt} = \frac{d\phi_w}{dy} \; at \; y = 0$$

Dynamic BC for water:
$$\frac{d\phi_w}{dt} + g * \eta = 0 \quad at \; y = 0$$

Dynamic BC for air:
$$u_{y,a} = \frac{d\phi_a}{dy} = \frac{d\phi_w}{dy} = u_{y,w} \quad at \; y = 0$$

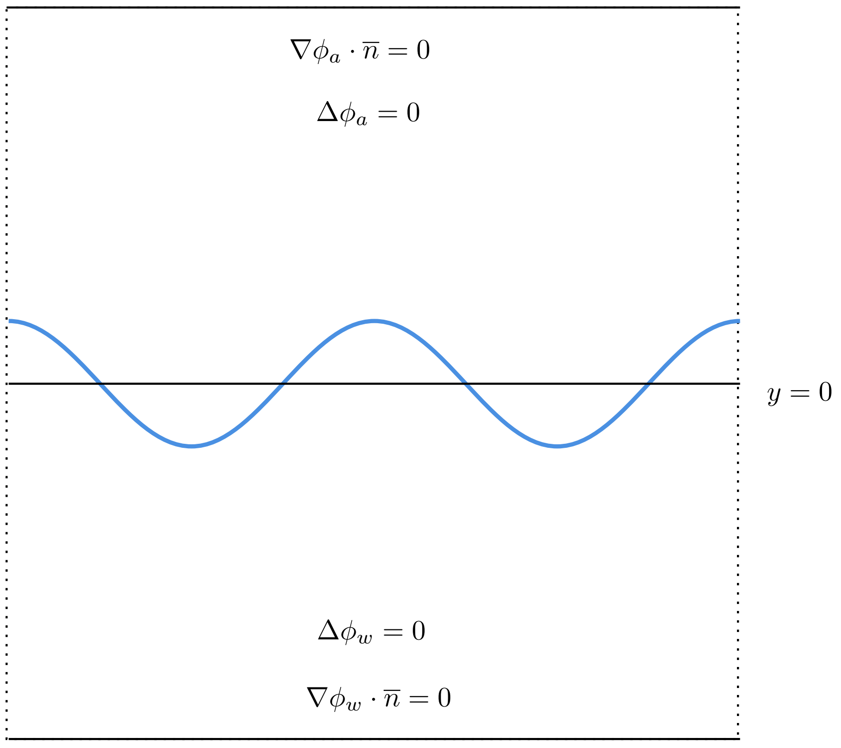

$\nabla\phi_a \cdot \overline{n} = 0$
$\Delta\phi_a = 0$
$y = 0$
$\Delta\phi_w = 0$
$\nabla\phi_w \cdot \overline{n} = 0$

Fig. 3.19 Wave problem particulars.

To see this, let us consider the first order wave problem using a potential flow approach along with a first order Stokes approximation. Meaning that the free-surface boundary conditions are applied at the mean free-surface position ($y = 0$) using a Taylor Series expansion and first order perturbation solutions [SG13b; Ser16]. Let us also formulate the corresponding problem for the air phase, therefore being the analytical solutions for the water and air phases:

$$\begin{gathered}\phi_W = A \cdot \frac{g}{\omega} \frac{\cosh K\,(H+y)}{\cosh K \cdot H} \cdot \cos\left(K \cdot x - \omega t + \delta\right) \\ \phi_A = -A \cdot \frac{g}{\omega} \frac{\cosh K\,(H-y)}{\cosh K \cdot H} \cdot \cos\left(K \cdot x - \omega t + \delta\right) \\ \eta = -A \cdot sin\left(K \cdot x - \omega t + \delta\right)\end{gathered} \tag{3.18}$$

Where A is the wave amplitude, K is the wave number, λ is the wave length, H is the water depth, ω is the wave frequency and g is the gravity. Both solution represent a monochromatic wave moving along the OX axis, but with different velocity fields. Figure 3.19 shows the solutions to obtain and the applied boundary conditions.

This can be interpreted as follows: The velocity obtained for air at the free surface has an opposite tangential velocity to the velocity obtained for water (Figure 3.20). This causes a velocity discontinuity to appear in the form of a slip boundary condition between fluid phases at the free surface which if not managed properly can cause interferences between the two fluids.

An approximated slip boundary condition for the velocity is imposed at the interface. This is achieved by interpolating the particle's velocities using only the nodal velocities of its fluid phase, and projecting only onto the same phase nodes.

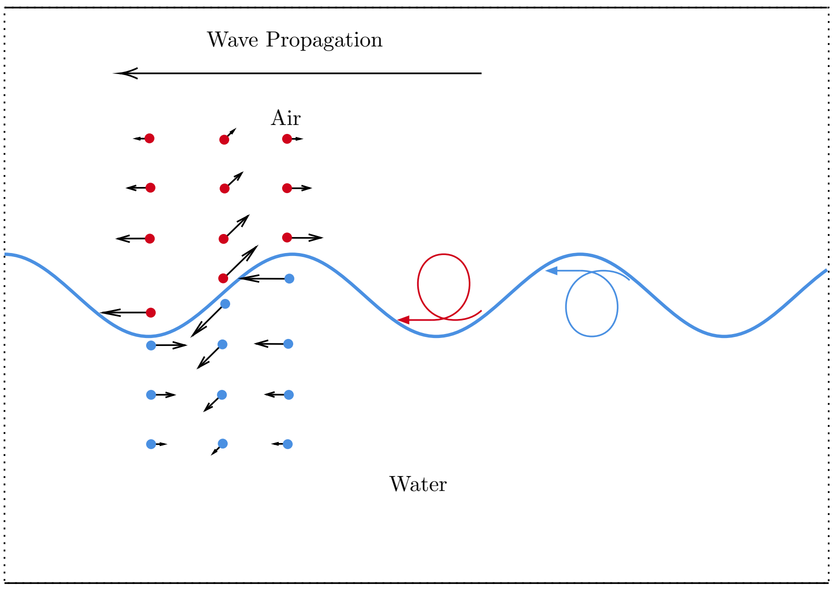


Fig. 3.20 Wave propagation and particle trajectory for air and water.

3.6 Validations and verifications

3.6.1 2D Standing wave problem

An initial validation for free surface flows can be done by comparing the standing wave problem in a closed tank. For small amplitudes and no viscosity we can compute an analytical result of the fluid movement from the potential flow theory. In fact, a good comparison is to compare first the free surface elevation and, second and most importantly, to compare the tank's forces and momentums that result from the movement of the fluid inside.

For this validation, we use a square one by one domain with the origin at the center and the center of gravity of the tank defined at the bottom where the forces and momentums will be computed:

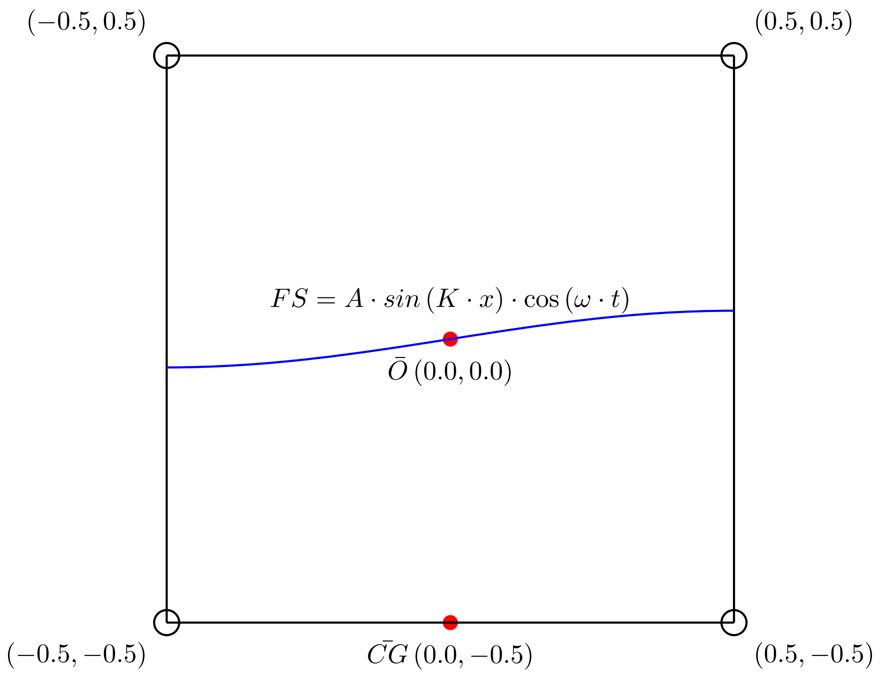


Fig. 3.21 Standing wave problem setup

As we can see, the free surface is defined using a sine function with an amplitude of $A = 0.01$. In this case, K is the wave number defined by $K = \frac{2 \cdot \pi}{\lambda}$ with $\lambda = 2m$ and ω is defined as the natural frequency of the tank obtained from the dispersion relation for first order Airy waves $\omega^2 = g \cdot K \tanh\,(K \cdot H)$ with H the water height $H = 0.5$ and g the gravity as $g = 9.80665$. So $K = \pi\ rad/m$ and $\omega \approx 5.36\ rad/s$.

3.6.1.1 Potential flow problem

For the standing wave problem calculated using the potential flow theory, we must consider two waves, one going in each direction. Therefore, the the potential for both waves and the free surface elevation are:

$$\begin{gathered}\phi\left(x,y,t\right)=2\cdot A\cdot\frac{g}{\omega}\frac{\cosh K\left(H+y\right)}{\cosh K\cdot H}\cdot\sin\left(K\cdot x\right)\cdot\cos\left(\omega\cdot t\right)\\ \eta=2\cdot A\cdot\sin\left(\omega\cdot t\right)\sin\left(K\cdot x\right)\end{gathered}\tag{3.19}$$

Integrating over the walls of the domain, the X-Axis force and the momentum are obtained analytically:

$$F_x=5725.88\cdot A\cdot\cos\left(\omega\cdot t\right)\tag{3.20}$$

$$M_z=-1229.84\cdot A\cdot\cos\left(\omega\cdot t\right)\tag{3.21}$$

These two equations will allow us to compare against numerically obtained results.

3.6.1.2 Numerical solution

The main focus of this analysis is to compare and see if the results obtained numerically are within the analytical results. The details of the problem are presented at table 3.1. Figure 3.21 shows the problem and the domain is discretised using a structured mesh. With the problem defined like this, we generate a mesh that is formed by 81 elements in each direction, obtaining a total of 13.122 elements (figure 3.22). In fact, a vertical refinement is applied so that the structured mesh is more refined close to the interface.

Viscosity	0.0
Water Density	$1000kg/m^3$
Simulation time	$10s$

Table 3.1 Standing wave simulation particulars.

Figures 3.23 and 3.24 shows the obtained pressure gradients and the velocity profiles obtained over the mesh. But to properly compare these results with the analytical ones, the X-axis forces and the Z-axis momentums must be compared.

Therefore, figure 3.25 presents the calculated forces and momentums compared with the analytical results obtained in the previous subsection. As it can be seen the response is a sinusoidal wave which compares with the analytical result but has a small decay factor the slowly decreases the amplitude of the forces/momentums.

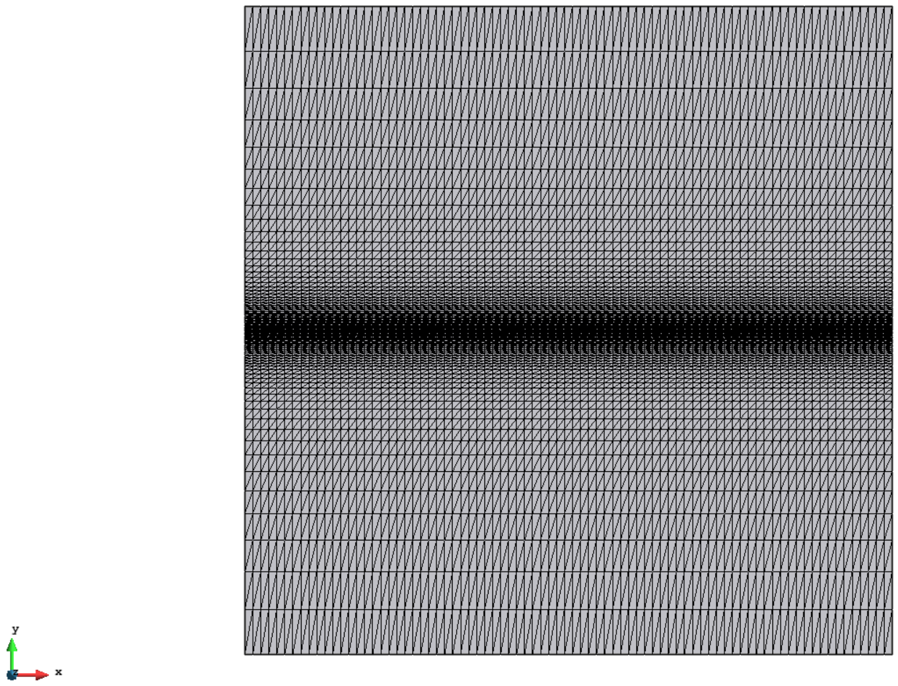


Fig. 3.22 Structured mesh obtained for the standing wave problem.

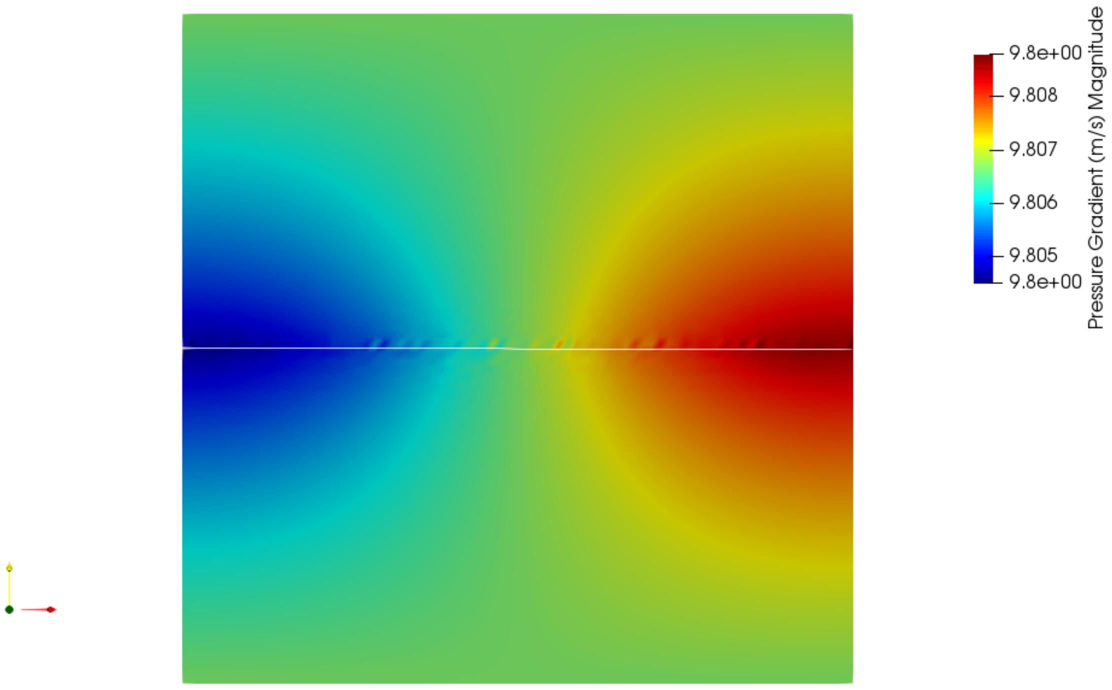


Fig. 3.23 Pressure Gradients obtained over the mesh at $t = 5s$

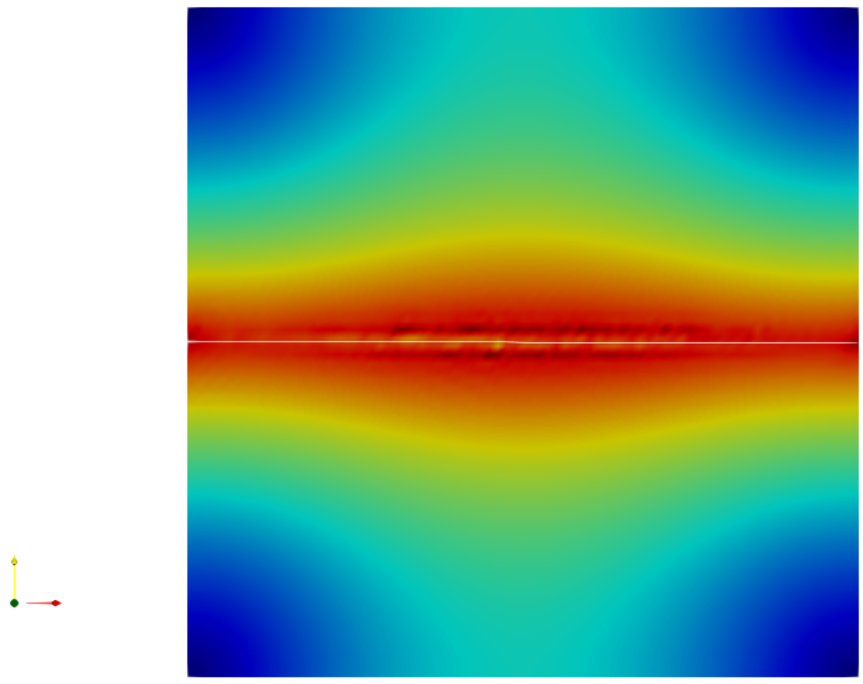
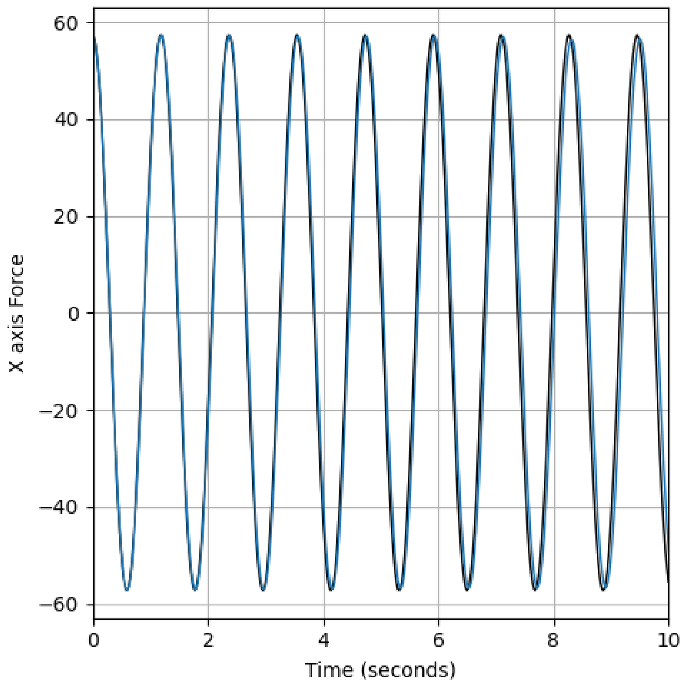
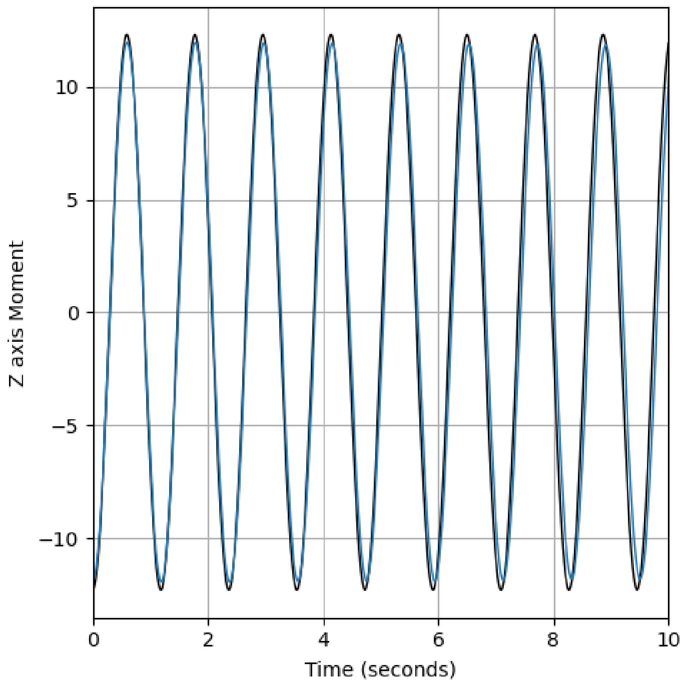


Fig. 3.24 Velocities obtained over the mesh at $t = 5s$



(a) Calculated X-axis forces(blue) compared with the analytical result(black)



(b) Calculated Z-axis momentum(blue) compared with the analytical result(black)

Fig. 3.25 Comparison of the calculated forces and momentums with the analytical results.

3.6.2 2D Sloshing problem

The focus of this validation case is to see how the proposed scheme copes with highly non-linear movements of a free surface problem inside a tank. In particular how capable it is of capturing the sloshing behaviour of a free surface flow inside a tank. The analysis is based on [Sou+06] where the sloshing behaviour is performed by obtaining the dynamic loads on the tank structure.

Capturing this behaviour is of high importance as liquid movement in tanks is a typical problem found in the naval industry. This phenomenon is important as the dynamic loads exerted on the walls can damage the ship structure or, worse, provoke large roll angle and even capsizing behaviour. In fact, the sloshing phenomenon can be used in favour to improve the ship's seakeeping capabilities when using Anti-Roll tanks which are often used to dampen the roll movement in ships.

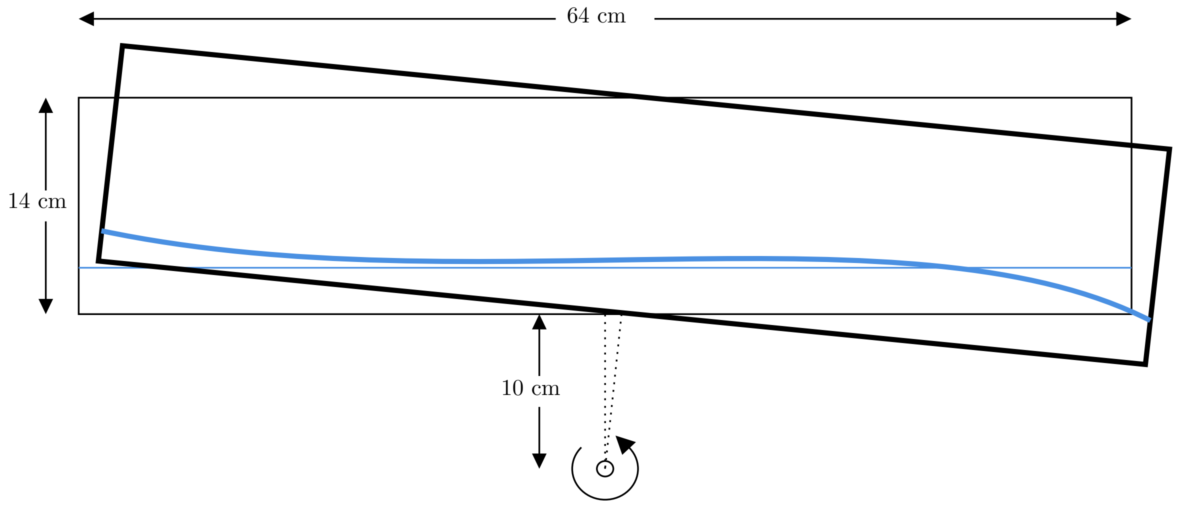


Fig. 3.26 2D Sloshing problem: Tank Geometry.

Reference [Sou+06] analyses the dynamic response of an anti-roll tank by forcing a rotation using an external momentum. This rotation is applied at different frequencies to see the maximum momentum amplitude and the phase differential between the dynamic response of the tank and the forced momentum. In particular, the proposed tank has a large jump in momentum amplitude when passing from frequency $4.34 rad/s$ to $4.87 rad/s$. This indicates that at high frequencies the fluid's motion cannot follow the forced motion and generates other waves that clash between themselves. This collision of the two waves prevents the high momentum amplitudes that appear for frequencies below $4.34 rad/s$.

The particulars of the tank are presented at figure 3.26 where the tank is forced to rotate around a point where the moment is measured. The rotation point is $10 cm$ below the tank's baseline and the maximum rotation applied is of $6deg$. The experimental campaign analysed the different response for frequencies that go from 0.24 to $8.21rad/s$ and with a water depth of $3cm$.

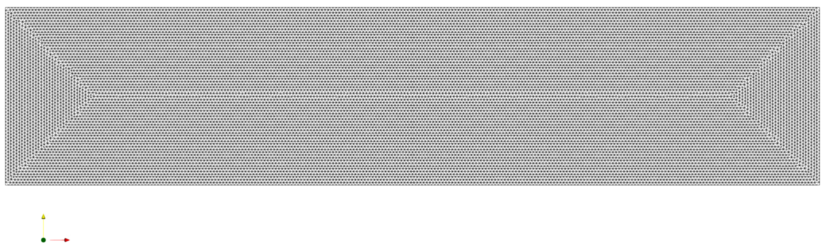


Fig. 3.27 Generated mesh for the 2D sloshing problem.

The experimental campaign shown at the reference [Sou+06] also shows a numerical result computed using an SPH. The numerical campaign used a total of 2541 particles and obtaining good and accurate results. Therefore, for this validation a relatively coarse mesh is used to obtain the results and compare against the SPH and the experimental results.

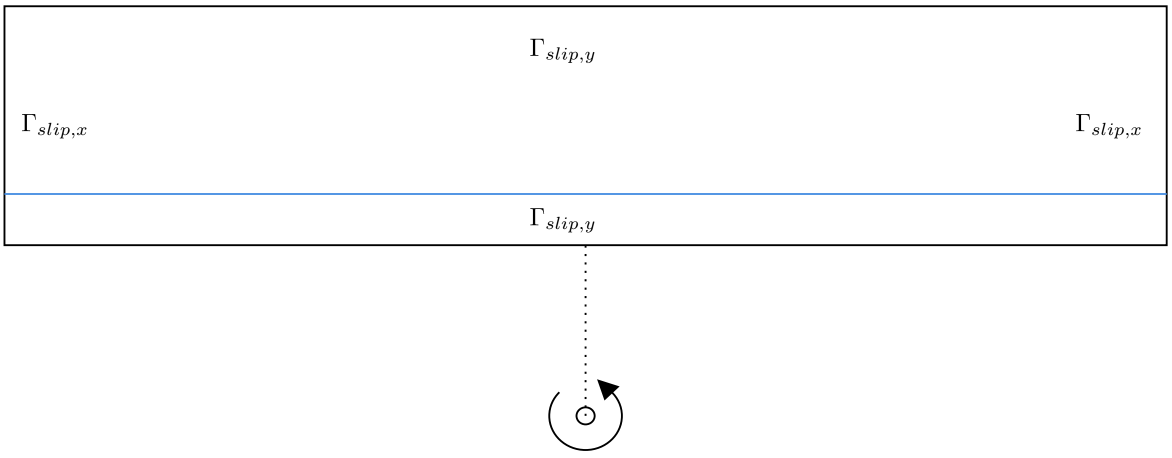
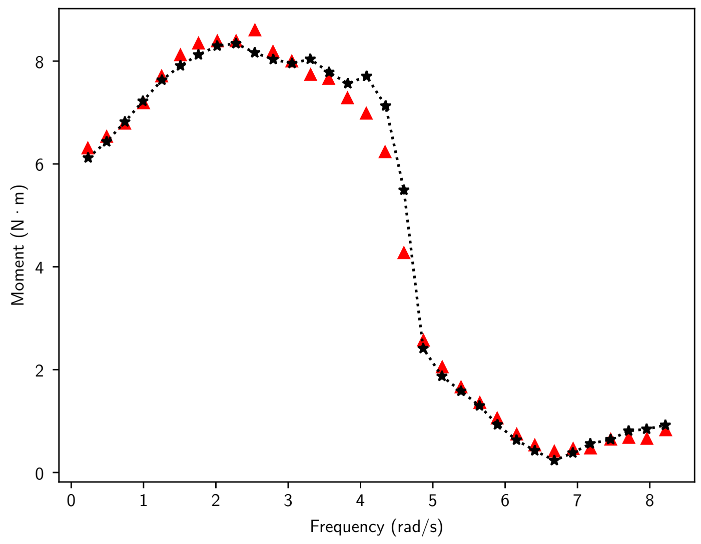
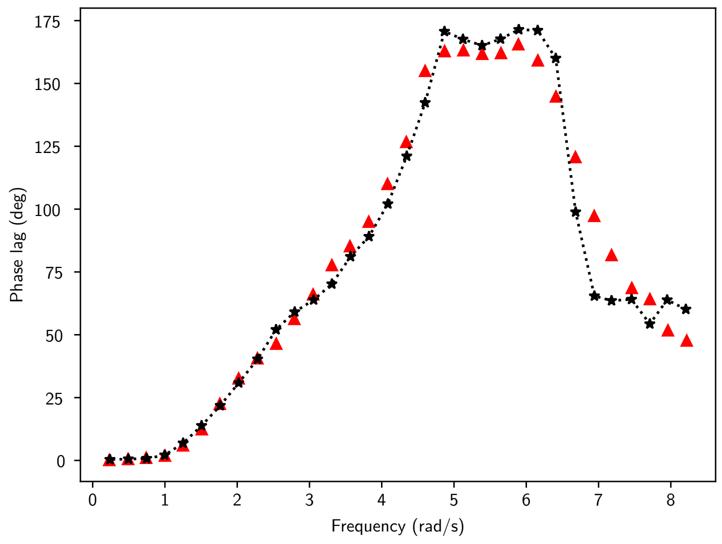


Fig. 3.28 2D Sloshing problem: Problem boundary conditions.

All of the simulations carried out use a mesh with an average size of $2.5 \cdot 10^{-3}\,m$ obtaining a total of 16.689 nodes and 32.752 elements. The mesh can be seen at figure 3.27 that shows the mesh used and figure 3.28 the boundary conditions applied to the problem (velocity slip conditions). The simulations are carried out using a time-step of $\Delta t = 2 \cdot 10^{-3}$, a total simulation time of $20\,s$ and a total of 10 particles per element.



(a) Fluid Moment Amplitude vs Frequency



(b) Moment Phase difference vs Frequency

Fig. 3.29 Fluid moment amplitude and phase difference with the rotation frequency comparison obtained experimentally (black) and numerically (red).

After running all of the cases for different rotation frequencies, a graph that shows the total moment amplitude and the phase difference between the fluid moment and the rotation is presented at figure 3.29. As it can be observed, good agreement is obtained between the experimental results and the ones obtained numerically. In particular the critical zone where the fluid moment passes from being in phase with the rotation to having a phase difference of $\pi\, rad$.

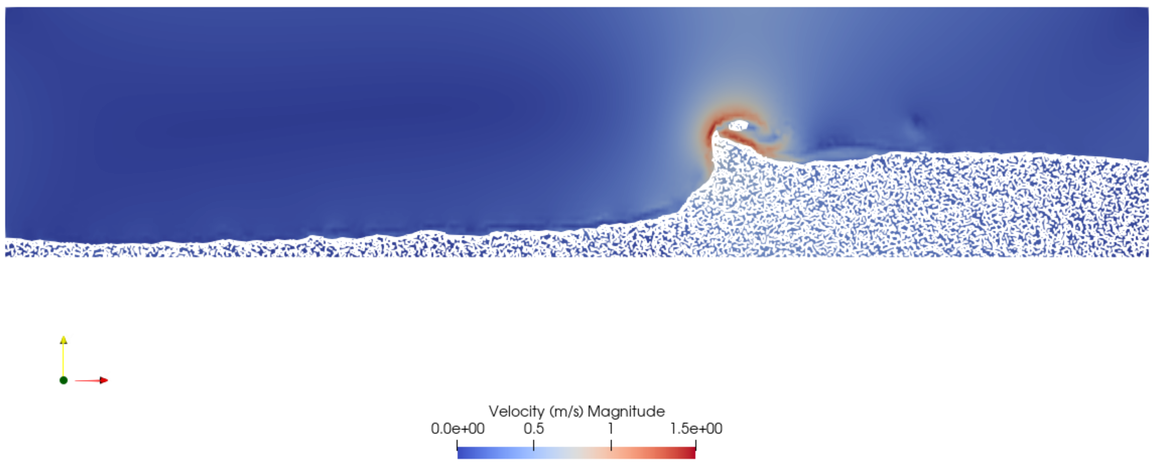
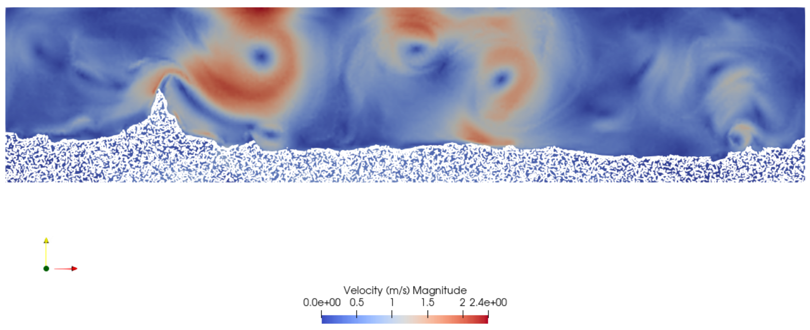


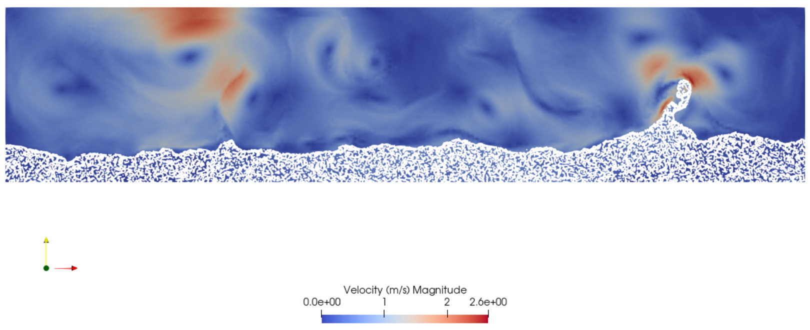
Fig. 3.30 Generated wave at the resonance condition.

When the fluid moment is in resonance with the rotation (frequency $4.34\, rad/s$), a large wave is generated that slams against the wall (see figure 3.30). But when the frequency is of $4.87\, rad/s$, then the fluid moment stops being in resonance as it generates more than one wave front that collide at the center of the tank, creating a positive effect which acts against the rotating moment. Figure 3.31 showcases two snapshots for a rotation frequency of $4.87\, rad/s$ which show the two wave collisions that appear.

Finally, videos that compare the obtained results agains the experimental campaign can be found at youtube by following the links at [Colb; Cola]. The videos show a superposition of the numerical particle's movement over the experimental video for the frequencies 4.34 and $4.87\, rad/s$.



(a) First wave collision



(b) Second wave collision

Fig. 3.31 Wave collisions produced for a frequency of 4.87 rad/s.

Case	Minim size	Number of Nodes	Number of Elements
1	0.02	13.000	71.000
2	0.01	69.000	425.000
3	0.00707	146.000	841.000
4	0.005	328.000	2.000.000

Table 3.2 Meshes used to compare the wave profile for a froude number of 0.316

3.6.3 Wave making resistance

This validation is based on the results obtained from the wave making resistance on a series 60 hull. The results are compared to the ones presented at the ITTC report [ITT85] where a variety of results are shown that come from different experimental campaigns.

The main difficulty of this validation is to properly catch the wave generated by the series 60 hull. Therefore, the level-set function to use in this case will be the smooth distance function as it allows for larger mesh-sizes without loosing accuracy. The specifics of the series 60 hull used in this validation is the same as the one used at [Ser16] with a hull length of $L = 1\,m$, breadth of $B = 0.130\,m$ and a block coefficient of $C_b = 0.6$.

In order to select the most adequate mesh, an analysis of the wave generation was performed for a froude number of 0.316 for different mesh sizes (shown at table 3.2). This convergence test is carried out maintaining a fix courant number, in order words, the time-step is reduced accordingly to the mesh size. The total simulation time is of 20 seconds which allows the wave to be generated correctly and, as figure 3.32 shows, the obtained forces obtained for case 2 have converged to a value. Also, it also shows how an initialisation of 5 seconds is added.

The wave pattern obtained over the hull is presented at figure 3.33 and shows that the all of the meshes obtain accurate results being the last two the ones that better approximate the wave pattern. Therefore, the second to last mesh is preferred over the last one as it is smaller and will require less computational time to solve. Figure 3.35 showcases the waves generated by the S60 for the chosen mesh.

Figure 3.34 presents the total resistance obtained at the different froude numbers (from 0.2 to 0.4 in 9 steps) compared with the experimental results obtained from [ITT85]. A few remarks must be made on how this wave making resistance is obtained.

From the experimental side, the wave making resistance is assumed to be independent from the viscous resistance therefore being able to compute the wave making resistance

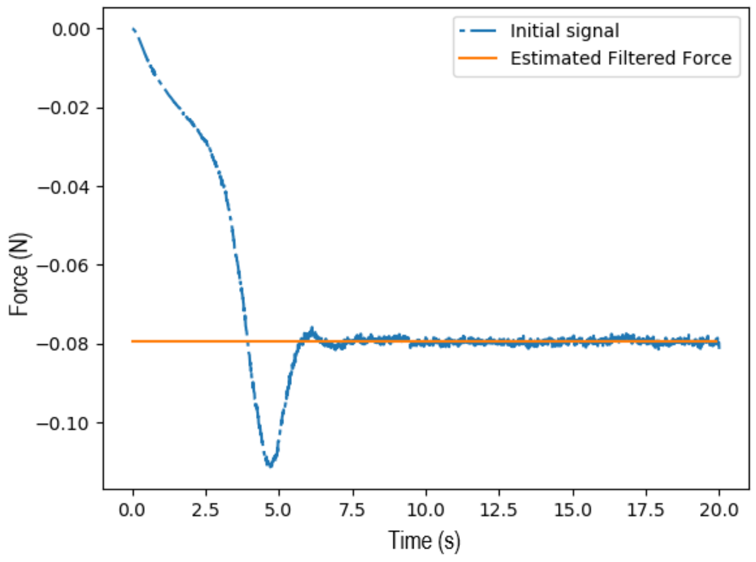


Fig. 3.32 X-Axis Pressure Forces obtained for case number 2

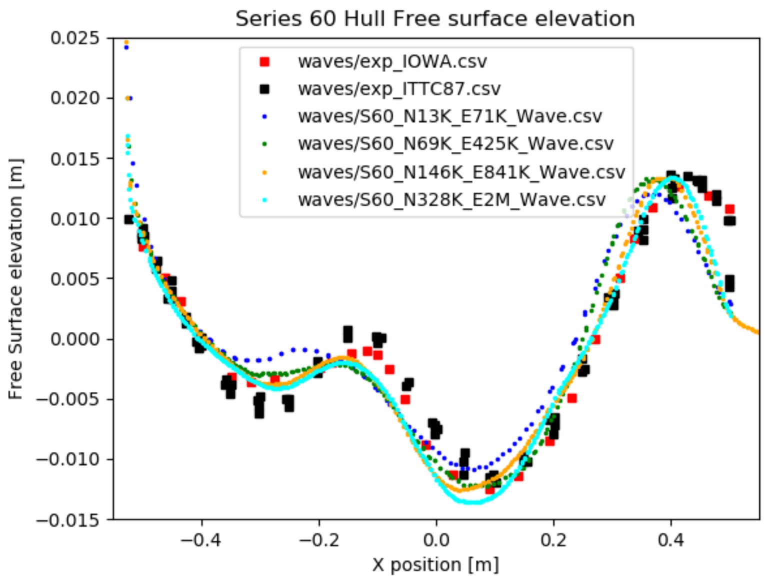


Fig. 3.33 Wave profiles for the different cases

as:

$$C_w = C_T - C_v \tag{3.22}$$

$$C_T = \frac{Resistance}{\frac{1}{2}\rho \mathbf{u}^2 A} \tag{3.23}$$

Being C_T the total resistance 3.23, C_v the viscous resistance and A the wetted area. So the main component that is missing is the viscous resistance. This component is approximated using the ITTC model-ship correlation line of 1957 3.24 and the form factor defined by Porhaska at [ITT75]. Equation 3.25, presents the equation used to obtain the wave making resistance.

$$C_F = \frac{0.075}{(log_1 0Re - 2)^2} \tag{3.24}$$

$$C_w = C_T - (1 + k)\, C_F \tag{3.25}$$

On the other hand, the numerical calculation of the resistance is performed by using a fluid with 0 viscosity. This assumption allows to compute the wave making resistance from the obtained pressure forces as:

$$C_w = \frac{Pressureforces}{\frac{1}{2}\rho \mathbf{u}^2 A} \tag{3.26}$$

The problem by assuming 0 viscosity is that in reality the numerical scheme does add some numerical diffusion that produces vorticity to appear and therefore increase the numerically computed wave resistance. Therefore, an overestimation of the wave making resistance is expected and is what is observed from figure 3.34.

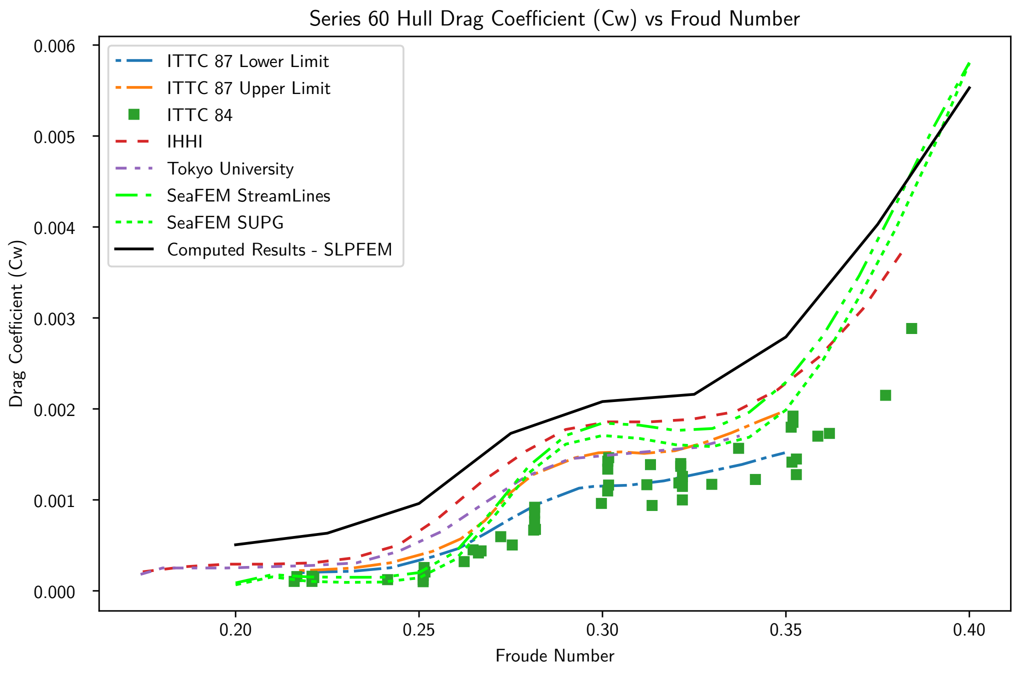


Fig. 3.34 Computed results compared to different experimental results and numerical results obtained using SeaFEM.

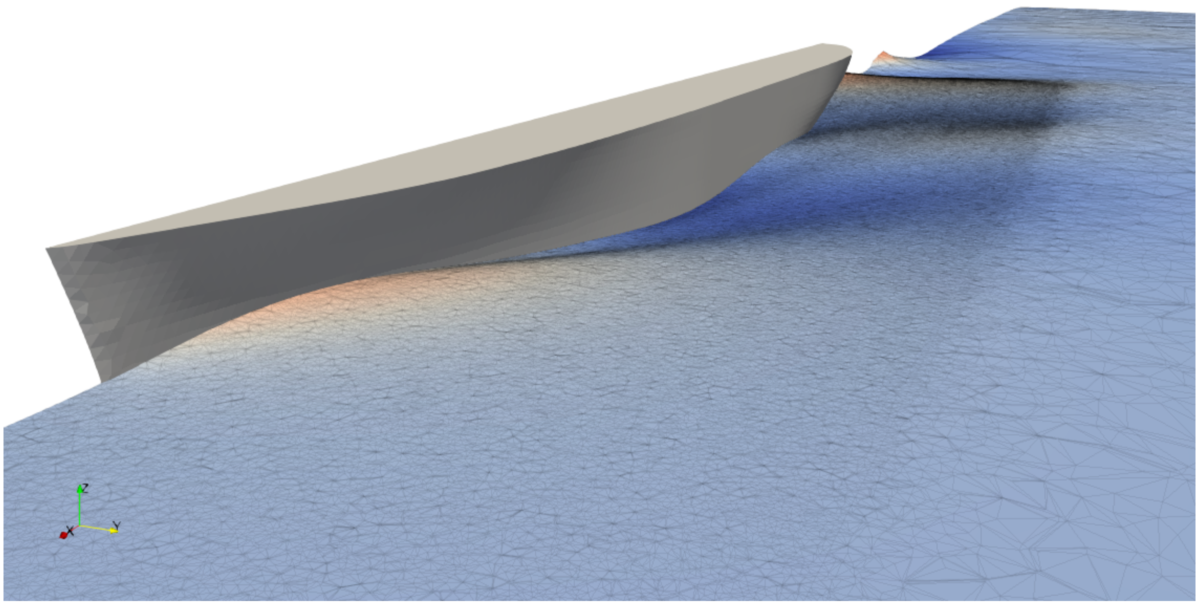


Fig. 3.35 Wave generation of the S60 hull for a Froude number of 0.316

3.6.4 Wave profile of a submerged cylinder

This validation observes the wave generation of a submerged cylinder when subjected to an inflow of fluid. The distribution of the problem is presented at figure 3.36, where the cylinder is submerged a total of 0.55 times the cylinder diameter. This analysis was thoroughly analysed at [GGF19] where different tests are performed by changing the Froude number and the Reynolds number to obtain the free-surface profile and the different modes obtained due to the vortex shedding of the cylinder.

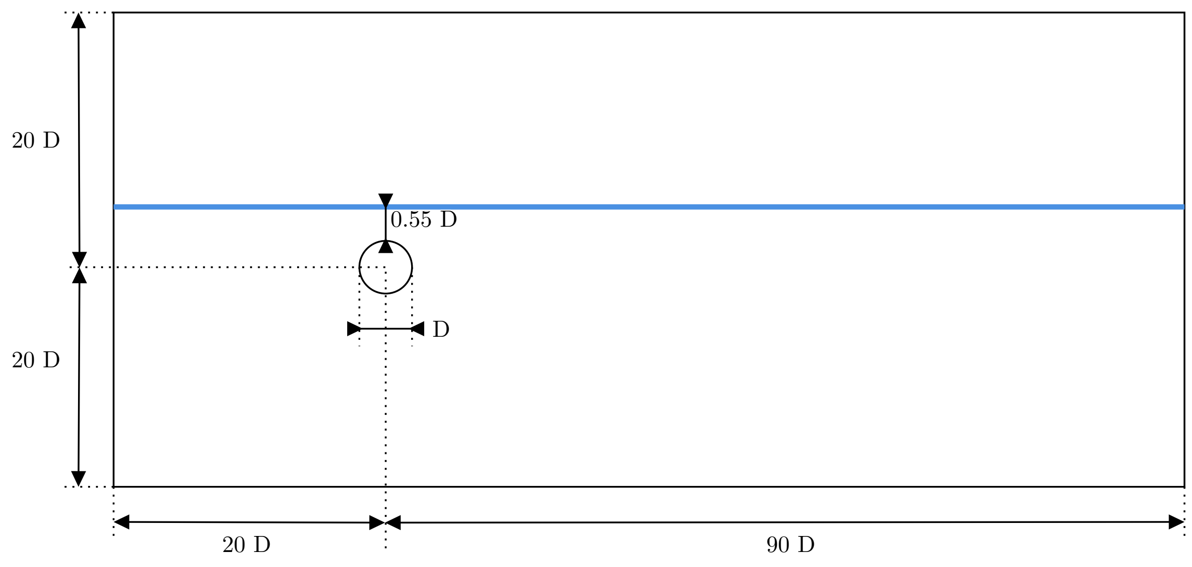


Fig. 3.36 Domain and cylinder distribution.

In this particular case, only one scenario is looked at where the wave profile is compared to a case for a Reynolds number of 40 and changing the Froude number. This case is compared against the results shown at [GGF19] to see if the obtained profile is comparable.

The case is defined using a structured mesh, mostly based on the one proposed at [GGF19] but instead of using quadrilateral elements, triangular elements are used. The domain is discretised using a structured mesh where the number of divisions are shown at figure 3.37, this way a total of 29.826 nodes are obtained and 58.642 triangular elements. The mesh can be seen at figure 3.38 and a close up to the cylinder is shown at figure 3.39.

The simulation is performed using different Froude numbers that vary from 0.5 to 3.0 in six steps (0.5, 1.0, 1.5, 2.0, 2.5 and 3.0). The simulation time-step is $\Delta t = 0.01\,s$ with a total simulation time of $t = 100\,s$. Also a total of 10 particles per element are used.

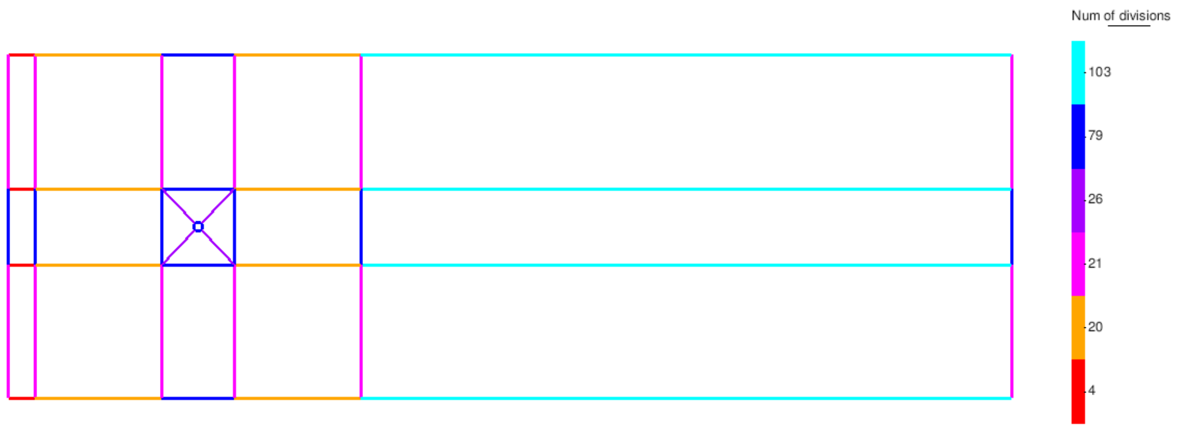


Fig. 3.37 Number of divisions to discretise the problem domain.

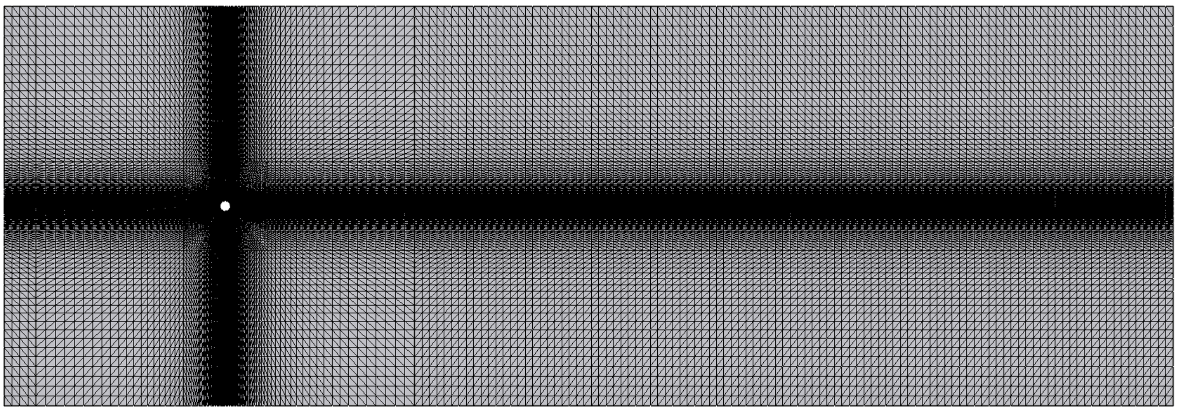


Fig. 3.38 Domain discretisation: Generated mesh

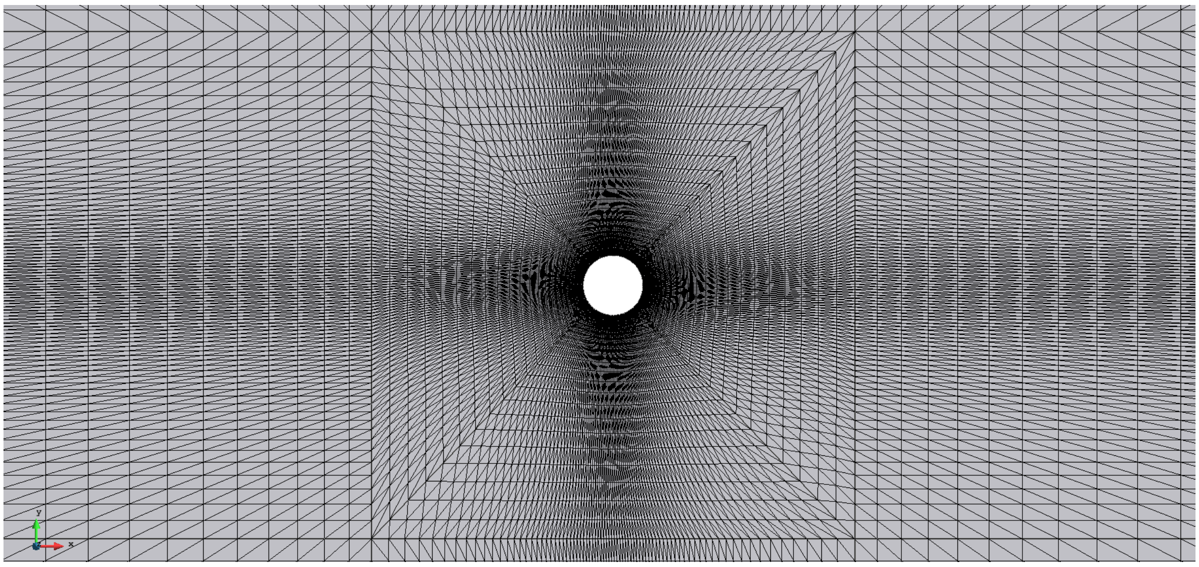
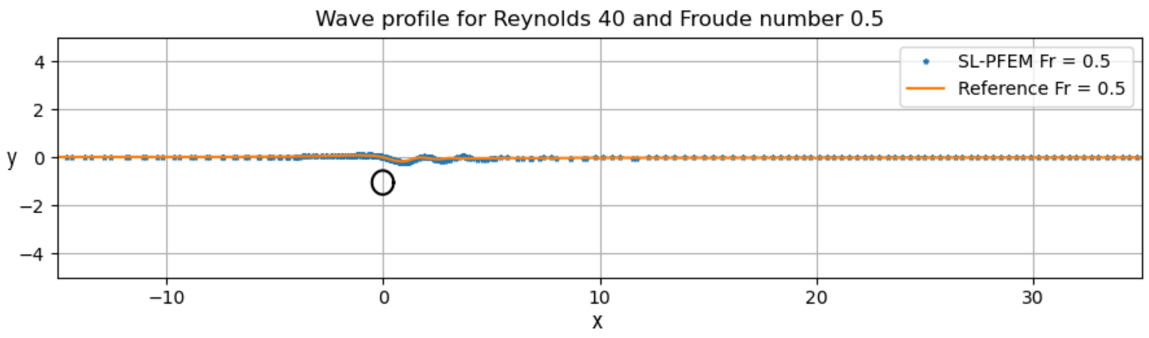
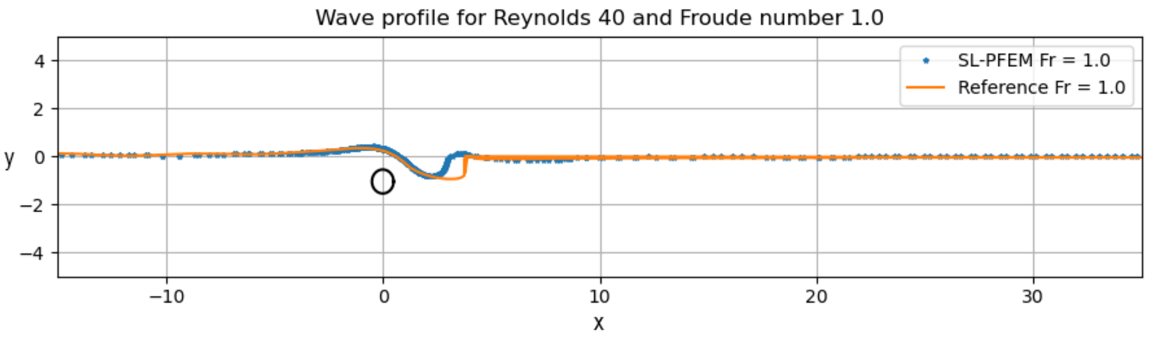


Fig. 3.39 Domain discretisation: Close-up of the generated mesh

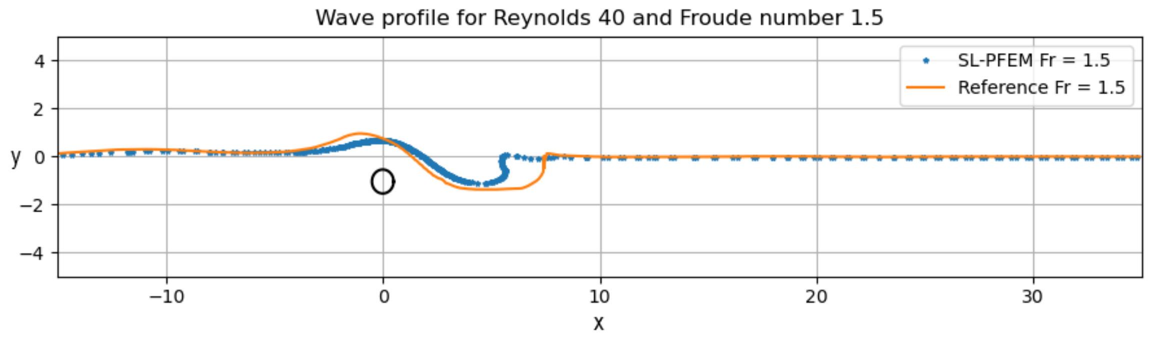
The results are presented at figure 3.40. Though differences are apparent the results do seem to be in line with the ones obtained at [GGF19]. In particular, the case for Froude 1.0 3.40b the wave generated is constantly breaking as can be seen at another snap-shot from 3.41 meaning that it is difficult to obtain a steady-state result.



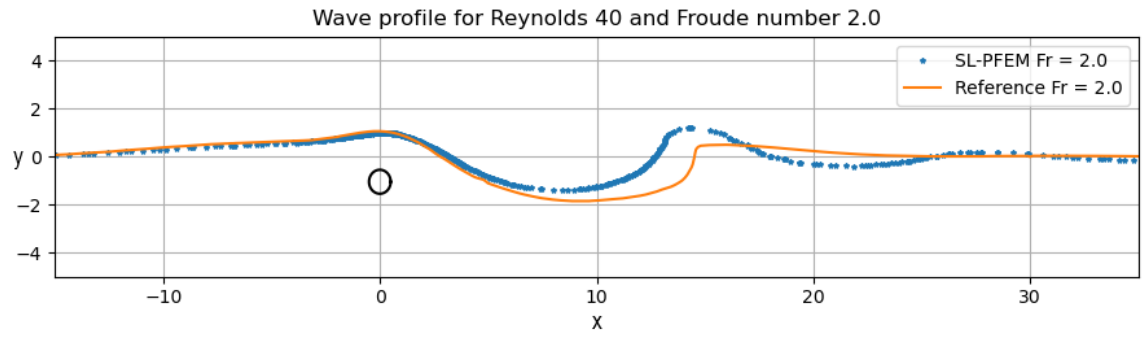
(a) Wave profile comparison for Froude number 0.5.



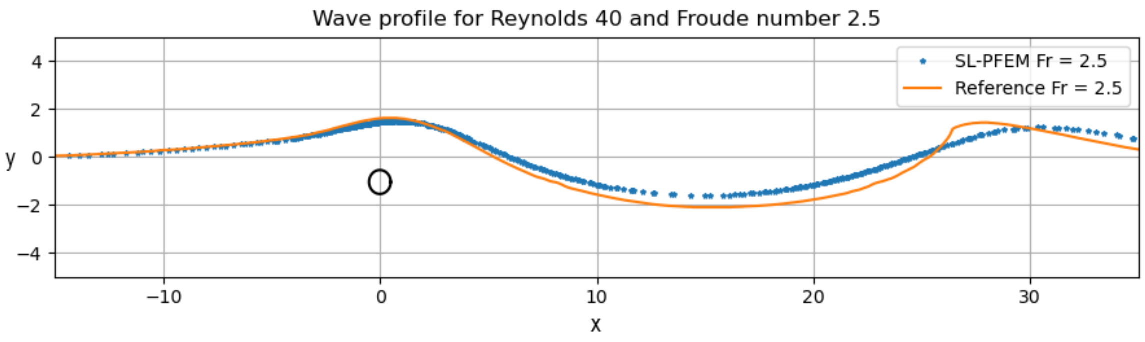
(b) Wave profile comparison for Froude number 1.0.



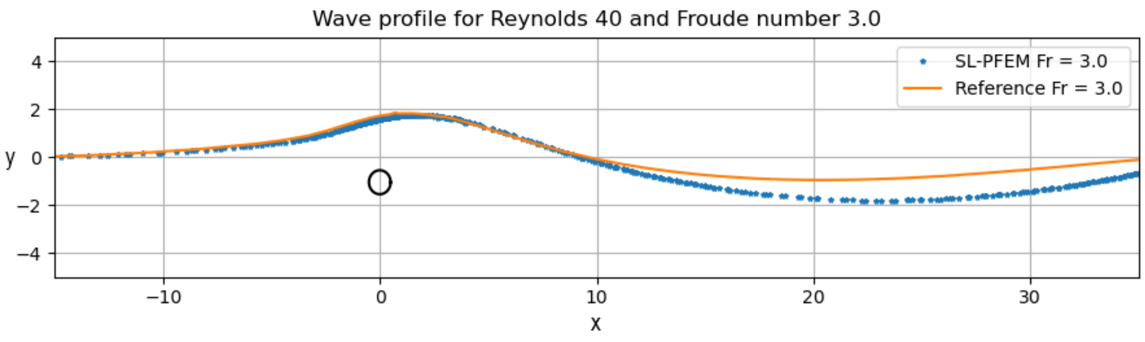
(c) Wave profile comparison for Froude number 1.5.



(d) Wave profile comparison for Froude number 2.0.



(e) Wave profile comparison for Froude number 2.5.



(f) Wave profile comparison for Froude number 3.0.

Fig. 3.40 Numerical wave profile (blue) compared to the results of [GGF19] (orange) for the different Froude numbers.

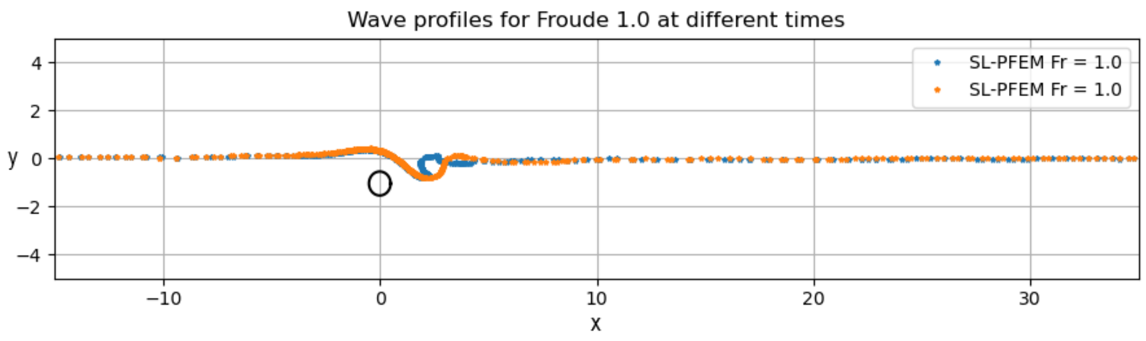


Fig. 3.41 Wave profile for Froude 1.0 at two different time-steps. Profile shown at figure 3.40b(orange) and at a different time step with a broken wave (blue).

Chapter 4

Fluid Solid problems

This chapter focuses on the simulation of moving objects within the fluid domain. Up to now, the development has focused on the interaction of a single fluid with different types of non-moving boundaries and on fluid-fluid interactions with the same types of boundaries. Therefore this last development is about implementing the possibility of having moving boundaries within the fluid domain.

These types of problems have been tackled in the passed using, for example, ALE (Aleatory Lagrangian-Eulerian) formulations [Oña+06] or the original PFEM [Ide+06a], [Oña+11]. The two formulations have one common drawback: time. The ALE formulation distorts the mesh in order to capture the moving boundary and therefore needs to rebuild the system's matrices within each time-step. PFEM, on the other hand, needs to first rebuild the mesh where deformations are high and then rebuild the system's matrices.

When enrichment is used to compute the solid-fluid interaction, then there is no need to re-mesh the domain or rebuild the system's matrices structures, only a local reassembly is needed. This approach highly improves the CPU time required to solve these types of problems. Therefore the ability to solve solid-fluid interactions in an efficient manner allows to compute simulations that have large amounts of solids. As stated at the objectives, the developments obtained here will be used to simulate ships navigating in ice, where on one hand we have the solid representing the ship and on the other hand we have the ice floes, all of these solids will be interacting with each other and with the fluid domain.

An example of a solid-fluid interface problem is shown at figure 4.1 where it represents a solid, in this case a cylinder, over the fluid mesh. Figure 4.2 shows the generated

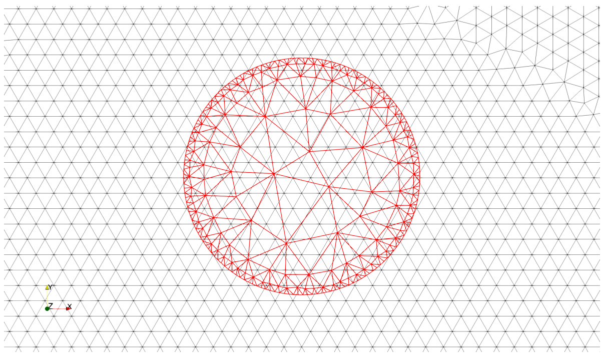
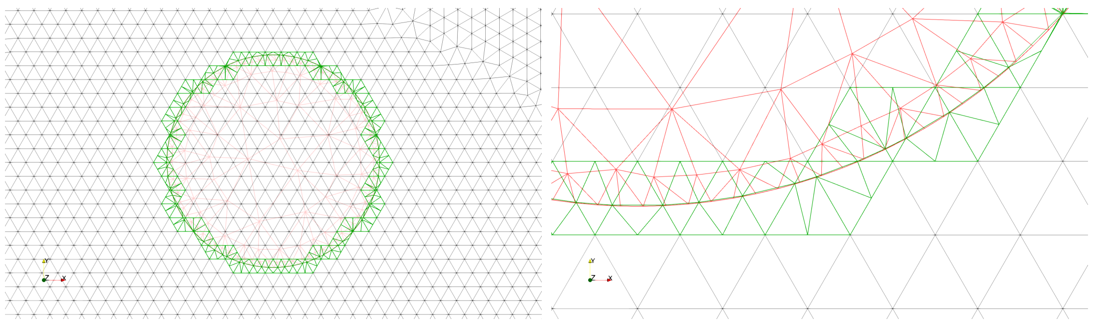


Fig. 4.1 Representation of the fluid domain mesh (black) and the solid using an independent mesh (red)

enriched elements around the solid in green. It can be observed that the mesh used for the cylinder (red) is refined at the contour. This is to accurately capture the interface over the fluid mesh.


(a) Enriched mesh

(b) Close-up of the enriched mesh

Fig. 4.2 Representation of the enriched mesh for the cylinder over the fluid domain.

Finally, an explicit algorithm is used to solve the solid-fluid interaction. Such schemes generally need a small time-step in order to obtain good results, but it is assumed that the fluid solver will be limiting the time-step and therefore will not be affected by the explicit coupling. The proposed explicit scheme is presented at algorithm ??, where the general algorithm that was presented at chapter 2 is extended in order to include the solid problem.

Algorithm 2: Full Algorithm for solving the enriched fluid-solid interactions

```
Compute Solid Intersections
Prepare Enriched systems
while (Iteration loop: ($P^{i+1}$ and $\hat{\mathbf{u}}^{i+1}$) !Converged)
 {
    Prepare General and Enriched Velocity Right Hand Side
    Solve Velocity $\hat{\mathbf{u}}^{i+1}$
    Prepare General and Enriched Pressure Right Hand Side
    Solve Pressure $P^{i+1}$
    Compute Pressure Gradients
 }
Interpolate Pressure from Mesh over solid interface
Body Forces: Integrate Pressure of the solid interface
Body Kinematics: Solid Solver computes and returns Body kinematics
```

4.1 Solid enrichment process

To begin the enrichment process, first the degrees of freedom where enrichment will be applied have to be defined:

- **Velocity**: The velocity degrees of freedom will be enriched in order to capture the discontinuity produced between the solid and the fluid. In fact, this will allow to apply different types of conditions.
- **Pressure**: The pressure degree of freedom will be enriched the same way as the free surface. This enriched variable is not needed when solving a solid interface as a pressure discontinuity does not appear at the solid interface, but it is implemented as it will be needed for multi-interface problems.

Algorithm ?? presents how the different generic steps to implement the enrichment process shown at the beginning of chapter 3 3.2 are inserted into the SL-PFEM method. In fact, the necessary steps to take can be seen at 4.1 where, in the same way as free-surface problems, some steps are a bit different due to the type of problem being solved and will be explained further on.

1. Find the intersections of the Solid interface with the mesh.
2. Divide the element into the local sub-elements.
3. Unassemble the original element from the main mesh.

4. Construct the local system of equations.
5. Collapse the local system into the global one.
6. Construct and collapse the enriched right hand side.

Find the intersections of the Solid interface with the mesh

To obtain the intersections of the solid mesh with the fluid mesh, two approaches can be taken:

- Use the Semi-Lagrangian Level-Set method to compute the intersections over the fluid mesh.
- Directly compute the intersections of the fluid-solid meshes.

Intersections using the SL Level-Set

The first scenario is the most simple way of obtaining the intersections. In this case, particles will contain information on whether they belong to a solid or not. Once they have been convected, this information is projected to the fluid mesh, allowing to obtain the intersections. This method is exactly the same as the one used to track the free-surface, but some problems arise from it.

The spatial accuracy comes from using the SemiLagrangian Level-Set method to obtain the interface. The level-set function to be used for these cases will be a step function, which at the previous chapter showed that an interface can be accurately captured with an appropriate number of particles. Therefore, the interface accuracy directly depends on the number of particles and the size of the fluid and solid meshes.

Mesh to Mesh intersection

The second approach is a more time-consuming method of obtaining intersections. In this case, the lagrangian frame is not used as a mesh to mesh computation of the intersections is performed. This methodology is more accurate tan the first one at the expense of higher computational time.

A brute force approach to compute the mesh to mesh intersections would be extremely time consuming. Let us imagine if a thousand of solid meshes are used with each solid mesh containing 200 elements. If the domain mesh contains 100.000 elements, then having to check each and every combination of element intersection would require a total of $2 \cdot 10^{10}$ element to element checks. The time it would take to perform this would be larger than the time to solve the fluid problem.

So the most important step is to simplify this by creating bounding boxes for each solid. Checking if an element is within the bounding box is very quick and this reduces by various factors the amount of element to element intersections to be performed.

Other problems that also appear is that nodes and particles need to check if they are within a solid mesh in order to apply the body dynamics to them. If this is not done using bounding boxes, it can prove to be an even more difficult job as all nodes and particles would have to check if they belong to a solid or not.

Still, even using bounding boxes, the fact of having to find the interface this way and having to check nodes and particles if they are in or out of a solid is very time-consuming but much more accurate than the aforementioned method. It allows to use larger fluid domain elements without loosing interface accuracy.

These two methods can be seen at figure 4.3 where in red the solid interface obtained using the first method is plotted over the background mesh (black. The second method is not plotted as it obtains the exact intersections of the solid object (blue). As it can be seen, the Semi-Lagrangian Level-Set method does not obtain a good solution compared to the original solid. Therefore, if precision is needed then the direct computation is preferred but if it is not the case, then the first method is more appropriate.

Construct and collapse the enriched right hand side

At the start of this section the two variables that will be enriched were defined as: pressure and velocity. Therefore, the right hand sides of each equation must be enriched. To do this, we can recall equation 3.15 that shows the local construction of the pressure system, but in this case, the same will have to be done for the velocity equation:

$$\begin{gathered}
\begin{bmatrix} (\mathbf{M} \bar{+} \nu\mathbf{K})_{kk} & (\mathbf{M} \bar{+} \nu\mathbf{K})_{k\star} \\ (\mathbf{M} \bar{+} \nu\mathbf{K})_{\star k} & (\mathbf{M} \bar{+} \nu\mathbf{K})_{\star\star} \end{bmatrix} \begin{bmatrix} \mathbf{u}_k \\ \mathbf{u}_\star \end{bmatrix} = \begin{bmatrix} b_{\mathbf{u}k} \\ b_{\mathbf{u}\star} \end{bmatrix} \\
\begin{bmatrix} \bar{\mathbf{K}}_{kk} & \bar{\mathbf{K}}_{k\star} \\ \bar{\mathbf{K}}_{\star k} & \bar{\mathbf{K}}_{\star\star} \end{bmatrix} \begin{bmatrix} P_k \\ P_\star \end{bmatrix} = \begin{bmatrix} b_{Pk} \\ b_{P\star} \end{bmatrix}
\end{gathered} \tag{4.1}$$

Also, as shown at equation 3.16, the enriched right hand side appears as part of the original system. Therefore, the enriched right hand side for the the different systems will

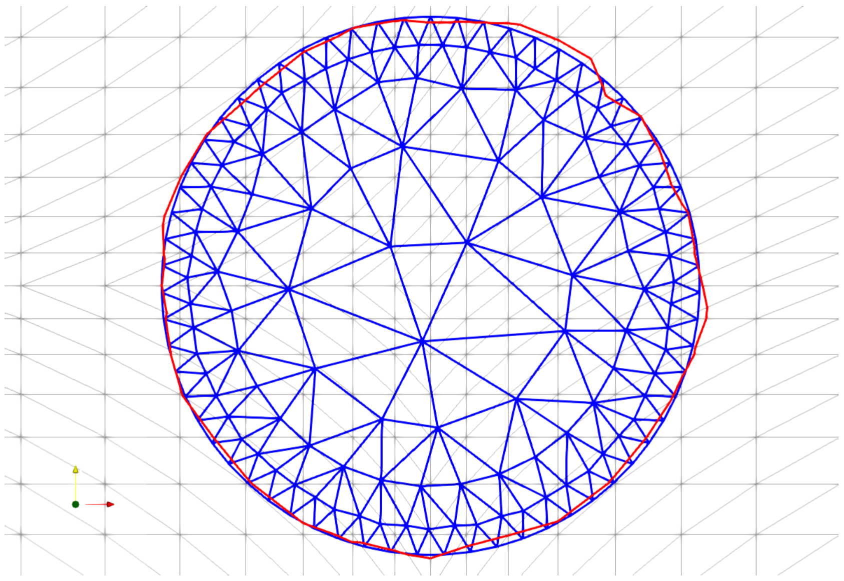


Fig. 4.3 Lagrangian projection of the solid interface (red) over the background mesh (black) compared to the solid object (blue)

be constructed in the same way as the original ones but, as in the free surface problems, minor differences appear:

- Compared to the free-surface problem (3.4.1), when computing the solid problem the velocity variable is being enriched. Therefore the enriched value at the local enriched node can be obtained and does not need to be interpolated.
- The pressure gradients have the same problem as expressed at 3.4.1 where if they are not interpolated the boundary integral of the equation is not properly compensated and can cause deterioration of the solution.

So, the final enriched right hand sides for each system can be written as:

$$
\begin{aligned}
b_{\mathbf{u}\star} &= \left[\bar{\mathbf{M}}_{\star k}\ \bar{\mathbf{M}}_{\star\star}\right] \begin{bmatrix} {\mathbf{u}^{n+1/2}}_k \\ {\mathbf{u}^{n+1/2}}_k \end{bmatrix} - \frac{\Delta t}{2} \Im \left(\bar{\mathbf{M}}_L^{-1} \frac{\bar{\mathbf{G}}}{\rho} {P^{i+1}}_k \right) \\
b_{P\star} &= -\frac{2}{\Delta t} \left[\bar{\mathbf{D}}_{\star k}\ \bar{\mathbf{D}}_{\star\star}\right] \begin{bmatrix} \hat{\mathbf{u}}_k^{n+1,i+1} \\ \hat{\mathbf{u}}_k^{n+1,i+1} \end{bmatrix} + \left[\bar{\mathbf{D}}_{k\star}^T\ \bar{\mathbf{D}}_{\star\star}^T\right] \begin{bmatrix} \bar{\mathbf{M}}_L^{-1} \frac{\bar{\mathbf{G}}}{\rho} {P^{i+1}}_k \\ \Im \left(\bar{\mathbf{M}}_L^{-1} \frac{\bar{\mathbf{G}}}{\rho} {P^{i+1}}_k \right) \end{bmatrix}
\end{aligned}
\tag{4.2}
$$

4.2 Rigid Body dynamics solver

The last part needed for this section is a tool that takes in the fluid forces and computes the body kinematics according to this input. Therefore, to compute the rigid body dynamics for the solids an external library is used as it has been already optimised for different scenarios:

- Rigid body dynamics solver for large displacements and rotations.
- Collisions between bodies.

The main reason to use an external library is that the main focus of this thesis is not the development of a rigid body dynamics solver. Especially considering that an optimised solution to compute large amounts of collisions that could be needed in new applications such as ice navigation.

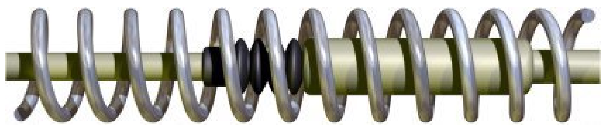


Fig. 4.4 Open Dynamics Engine. Source [Smi].

The Open Dynamics Engine [Smi] (from now on ODE) is the chosen solver for body dynamics that has been worked on quite extensively and has proven to work well with unstructured meshes. In fact, this thesis uses two of the main utilities: The collision detection and the body dynamics solver.

The only problem with ODE and any of the known solid dynamics solvers that have collision detection incorporated is that the main use is for computer graphics. As these problems do not need to be physically accurate, simplifications are done in order to obtain extremely fast and visually attractive simulations.

ODE does not give much information on the temporal scheme used, but it does indicate at the documentation that if physically accurate results are needed then a small time-step is recommended. This brought up some questions on what type of scheme it uses, and after some effort in reading the source code, the scheme used was found to be a simple explicit forward euler.

This scheme when coupled with the SL-PFEM method in an explicit manner using large time-steps can be really problematic for the physical accuracy of the solid solver ODE. Furthermore, if large time-steps for ODE are chosen then the collision detection can fail or allow for solid overlap. These internal problems from ODE are solved by implementing a sub-stepping method for the body dynamics solver. This brings more stability to the scheme and allows ODE to give out physically accurate results without affecting the total compute time as ODE takes very little time to compute collisions and body dynamics.

In conclusion, once the sub-stepping is implemented, the algorithm used is transformed as shown at algorithm 3.

Algorithm 3: Full Algorithm for solving the enriched solid-fluid interactions

```
Compute Solid Intersections
Prepare Enriched systems
while ($Iteration\ loop:\ (P^{i+1}\ and\ \hat{\mathbf{u}}^{i+1})\ !Converged$)
 {
    Prepare General and Enriched Velocity Right Hand Side
    Solve Velocity $\hat{\mathbf{u}}^{i+1}$
    Prepare General and Enriched Pressure Right Hand Side
    Solve Pressure $P^{i+1}$
    Compute Pressure Gradients
 }

for ($Sub\text{-}Stepping\ loop:\ 1\ to\ n\,sub-steps$)
 {
    Interpolate Pressure from Mesh over solid interface
    Body Forces: Integrate Pressure of the solid interface
    Body Kinematics: Solid Solver computes and returns Body kinematics
 }
```

4.3 Multi-interface enrichment: Simultaneous Fluid-Fluid and Solid-Fluid Interfaces

The final step of the thesis is to obtain a simulation tool that is capable of solving fluid-fluid (free surface) problems and solid-fluid problems at the same time.

Up to now, the two interface problems have been presented and analysed in a separate manner using the proposed enrichment method. In fact, as stated at 3.2, the selected enriched elements allow to take into account multiple intersections in order to compute a variety of interfaces being, for this problem, the fluid-fluid and the solid-fluid the two interfaces.

In this case, the two enrichment processes are combined into one, allowing to compute at each element all of the enriched variables.

4.3.1 Multi-interface enrichment process

To begin the enrichment process, first the degrees of freedom where enrichment will be applied have to be defined:

- **Velocity**: This variable is inherited from the solid-fluid interface problems where the velocity degrees of freedom are enriched in order to capture the discontinuity produced between the solid and the fluid.
- **Pressure**: This variables is inherited from the free surface interface problem. The pressure degree of freedom will be enriched the same way as the free surface. The big advantage is that even if for solids it was not needed, by having it already implemented it allowed for a fast implementation of multi-interface problems.

Therefore, the necessary steps to take to enrich the problem can be seen at 4.3.1 where, in the same way as the previous problems, some of the steps are combined or modified in order to accept the two interfaces.

1. Find the intersections of the different interfaces with the mesh.
2. Divide the element into the local sub-elements.
3. Unassemble the original element from the main mesh.
4. Construct the local system of equations.
5. Collapse the local system into the global one.

6. Construct and collapse the enriched right hand side.

As with the two previous enrichment methods, some of the points do not need further explanation as the methodology is the same as the previously presented ones, but some other steps do.

Find the intersections of the Solid interface with the mesh

When tackling the multi-interface problem, two or more interfaces are available. In particular one free surface interface and various solid-fluid interfaces. Therefore the easiest way to proceed is by using the methods proposed for each problem in particular: For free surfaces use the semi-Lagrangian Level-Set method to obtain the interface at the mesh and, for solids, use any of the two methods proposed before to obtain the solid interfaces at the mesh. The advantage is that it can be done independently.

Figure 4.5 presents how in one given element two interfaces can appear.

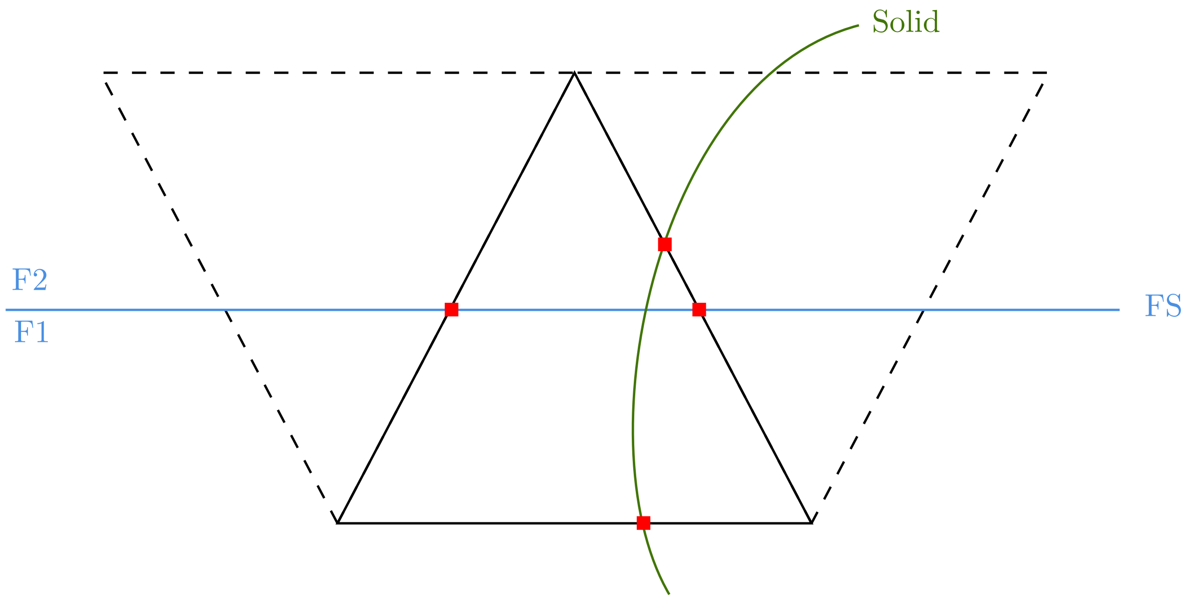


Fig. 4.5 Multi interface problems. Interfaces that can appear at one triangle.

Divide the element into the local sub-elements

Up to now, this step was exactly the same as the step proposed at the generic enrichment process of chapter 3. For multi-interface problems, this is still the same but one minor simplification must be added. Figure 4.5 perfectly shows the problem that appears when using multi-interface enrichment where one of the sides of the triangle is intersected by two interfaces. This has to be solved as the enriched element only contains four sub-elements.

The solution to this is analysing the problem from an error perspective. The solid interface needs to keep its shape without changing as the interface is constantly known meanwhile the free surface is computed using the SemiLagrangian Level-Set method.So, distorting the free surface interface by moving the free surface intersection of the double intersected edge to the solid intersection solves the problem. Figure 4.6 shows how the sub-elements are created after performing this correction on the example shown at 4.5.

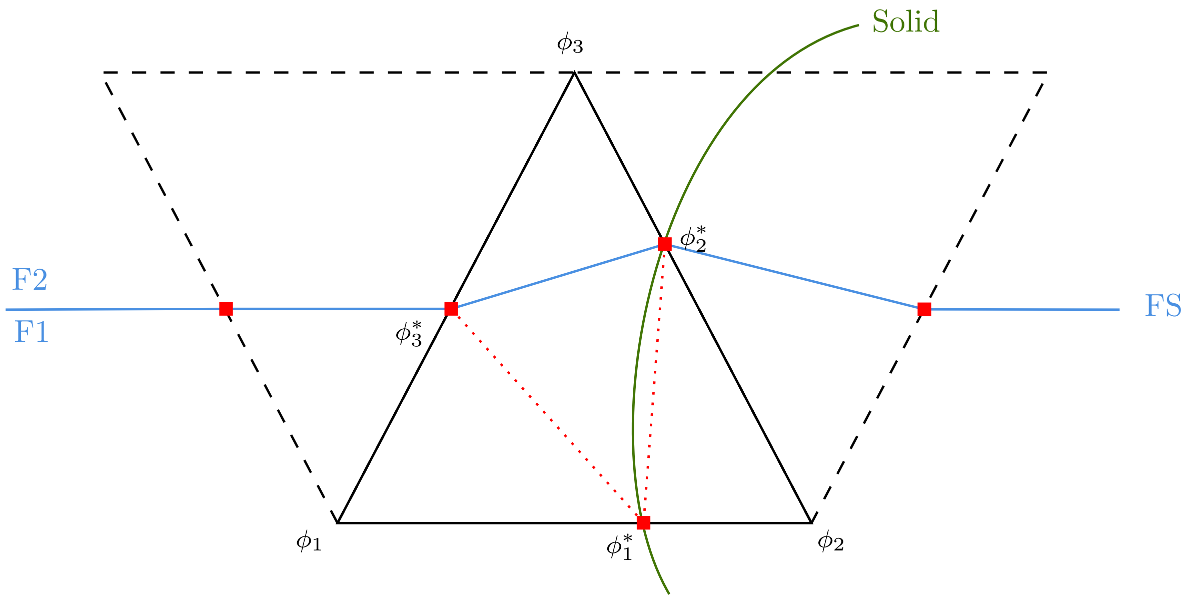


Fig. 4.6 Multi interface problems. Intersection correction and sub-element division.

Construct and collapse the enriched right hand side

This final step is exactly the same as the one presented at the solid enrichment process. The problem though is that enrichment is applied to the velocity variable only for solid-fluid interface and, to the pressure variable for both interfaces. Therefore special care must be taken when assembling the problems as some matrices have to be enriched for fluid-solid interfaces only, fluid-fluid interfaces only or for both types of interfaces at the same time.

Taking into account the pressure gradient problem presented previously 3.4.1, then the enriched right hand sides are presented at 4.3

$$
\begin{aligned}
b_{\mathbf{u}\star} &= \left[\bar{\mathbf{M}}_{\star k}\ \bar{\mathbf{M}}_{\star\star}\right] \begin{bmatrix} \mathbf{u}^{n+1/2}{}_{k} \\ \mathbf{u}^{n+1/2}{}_{\star} \end{bmatrix} - \frac{\Delta t}{2}\Im\left(\bar{\mathbf{M}}_{L}^{-1}\frac{\bar{\mathbf{G}}}{\rho}P^{i+1}{}_{k}\right) \\
b_{\star} &= -\frac{2}{\Delta t}\left[\bar{\mathbf{D}}_{\star k}\ \bar{\mathbf{D}}_{\star\star}\right] \begin{bmatrix} \hat{\mathbf{u}}_{k}^{n+1,i+1} \\ \hat{\mathbf{u}}_{\star}^{n+1,i+1} \end{bmatrix} + \left[\bar{\mathbf{D}}^{T}_{k\star}\ \bar{\mathbf{D}}^{T}_{\star\star}\right] \begin{bmatrix} \bar{\mathbf{M}}_{L}^{-1}\frac{\bar{\mathbf{G}}}{\rho_f}P_f^{i+1}{}_{k} \\ \Im\left(\bar{\mathbf{M}}_{L}^{-1}\frac{\bar{\mathbf{G}}}{\rho_f}P_f^{i+1}{}_{k}\right) \end{bmatrix}
\end{aligned}
\tag{4.3}
$$

4.3.1.1 3D tetrahedron enrichment

This last sub-section shows how a 3D tetrahedron is split when it is enriched by the two interfaces, as shown at figure 4.7. It is very similar to the 2D case where the division is done by using eight sub-elements instead of four: Four nodal ones and four central ones.

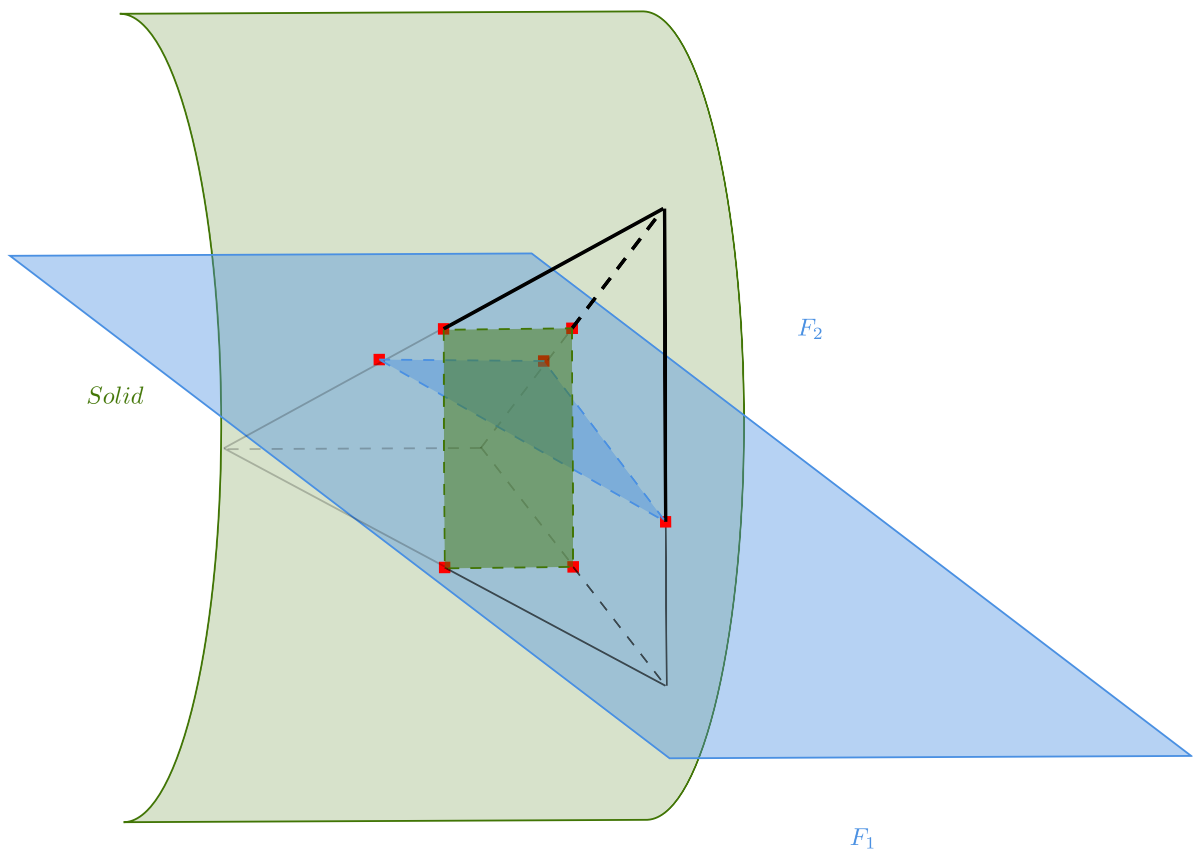


Fig. 4.7 Initial intersections of the interfaces with a single tretrahedron

Again, when the two interfaces are found, the edges are intersected by more than one interface, therefore the free-surface is modified in order to coincide with the solid

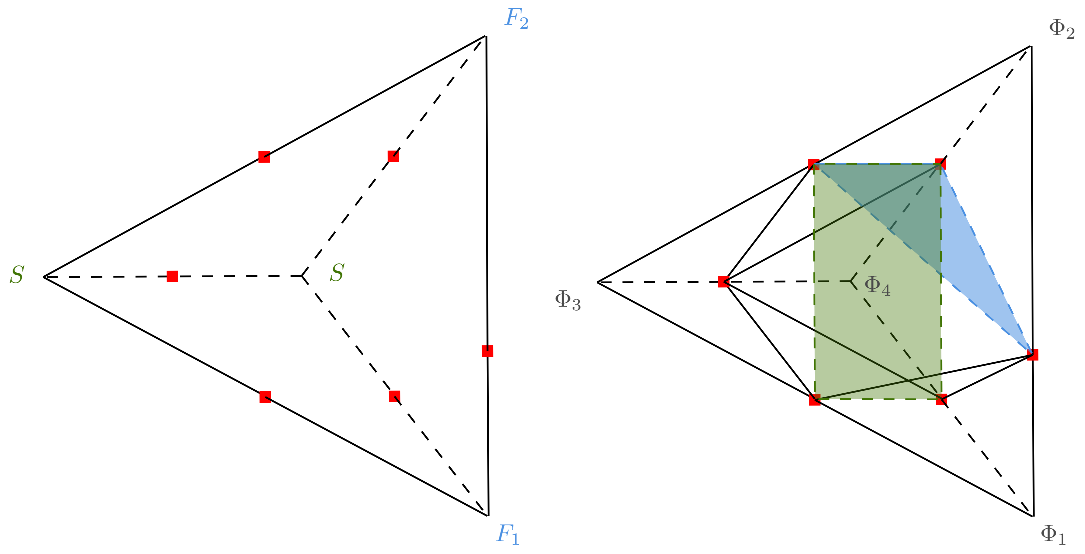


Fig. 4.8 Construction of the sub-elements within the tetrahedron

intersections, see figure 4.8 . The final division is presented at figure 4.9, where all of the eight sub-elements are shown and coloured depending if they belong to the fluid domain or to the solid domain.

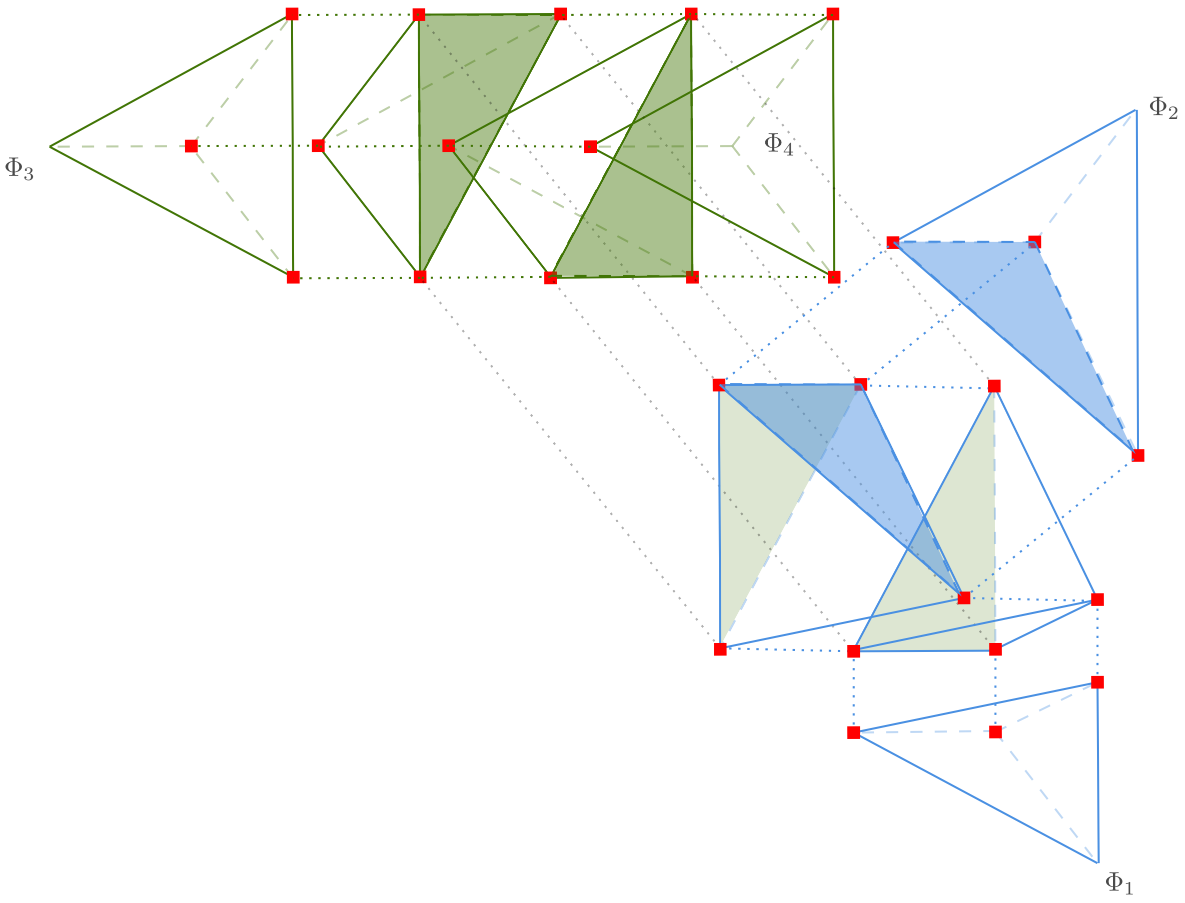


Fig. 4.9 Final division of the sub-elements within the tetrahedron

4.4 Validations and verifications

4.4.1 2D flow around a cylinder - Mesh Convergence

This first validation is to check the rate of convergence when using the solid enrichment process. To perform the validation a 2D case that computes the flow around a cylinder is used in a similar fashion to the sixth validation of chapter 2 section 2.6.8. The idea is to check the convergence rate of the numerical strouhal number.

The particulars of this problem are shown at table 4.1. As it can be observed, the fluid domain close to the cylinder is formed by a circular surface. This is done in order to generate a structured mesh that can be reduced in size in order to perform the mesh convergence analysis. In fact, it also allows to generate the mesh so that nodes do not coincide with the cylinder.

Cylinder diameter (D)	$1\,m$
Cylinder height (H)	$1\,m$
Inlet Velocity (V)	$1\,m/s$
Viscosity (ν)	0.005
Re	200

Table 4.1 Flow past a cylinder particulars

Figure 4.10 presents the domain used. It shows that the domain is large enough to allow for the development of the vortexes. Figure 4.11 and 4.12 present the different meshes used with the cylinder in place being the latter figure a close-up of said meshes. It can be observed at 4.12 that the structured meshes do not coincide with the cylinder mesh. The data of the meshes used can be seen at table 4.2.

Figure 4.13 presents the rate of convergence of the analysis. This figure shows the error for each mesh in a logarithmic scale and a tendency line to see the rate of convergence. In fact, the convergence rate shows that is second order (tendency line) of this validation case. The results are also shown at table 4.3. As a final remark, to obtain the vortex shedding frequency an FFT analysis is performed on the lift forces output.

Finally, figures 4.14 and 4.15 show the velocity magnitude and the pressure gradient for each of the cases ordered from top to bottom.

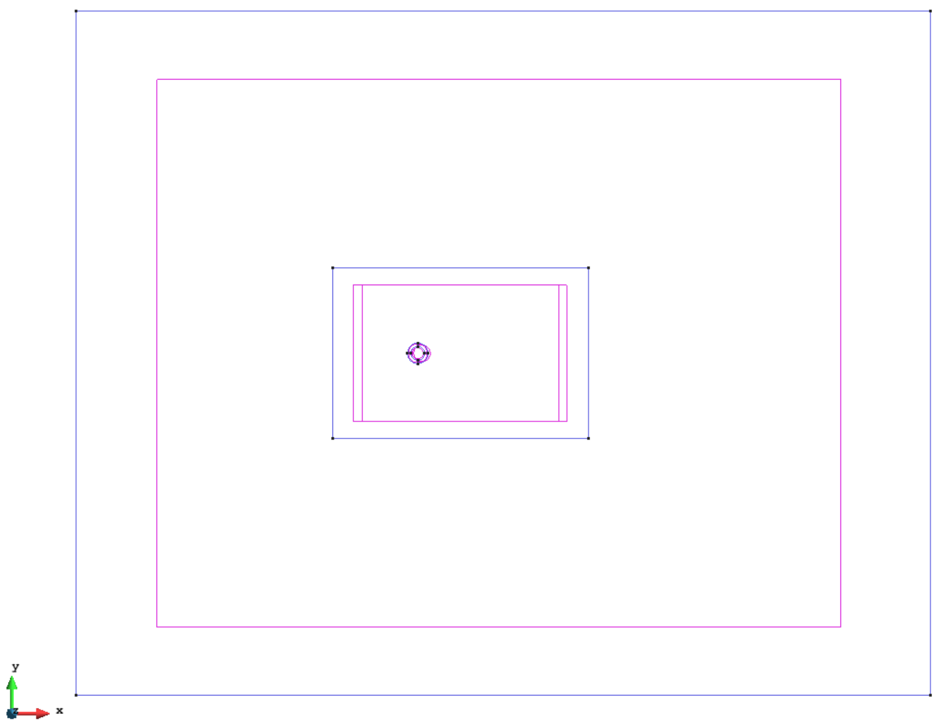


Fig. 4.10 Domain used for the 2D flow around a solid cylinder.

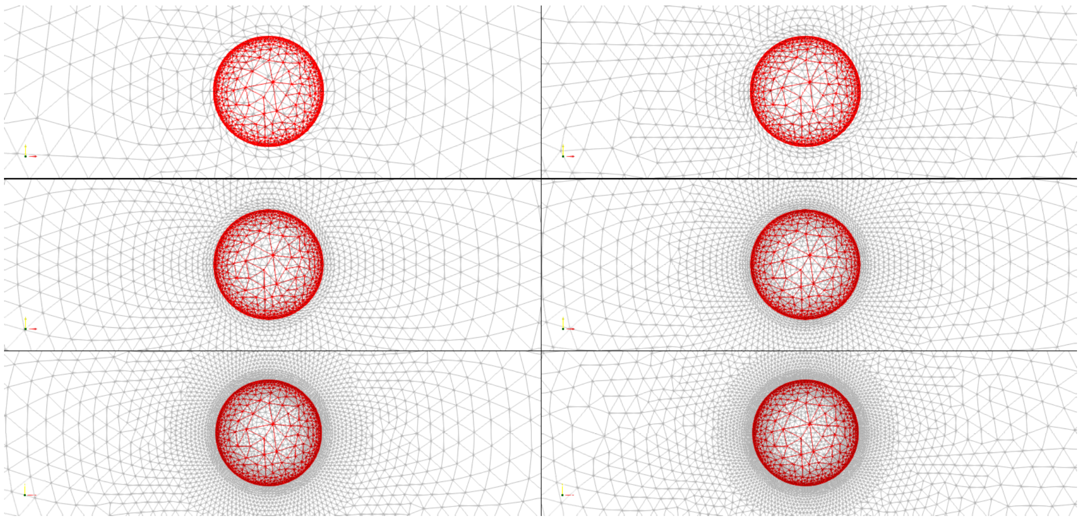


Fig. 4.11 Domain used for the 2D flow around a solid cylinder.

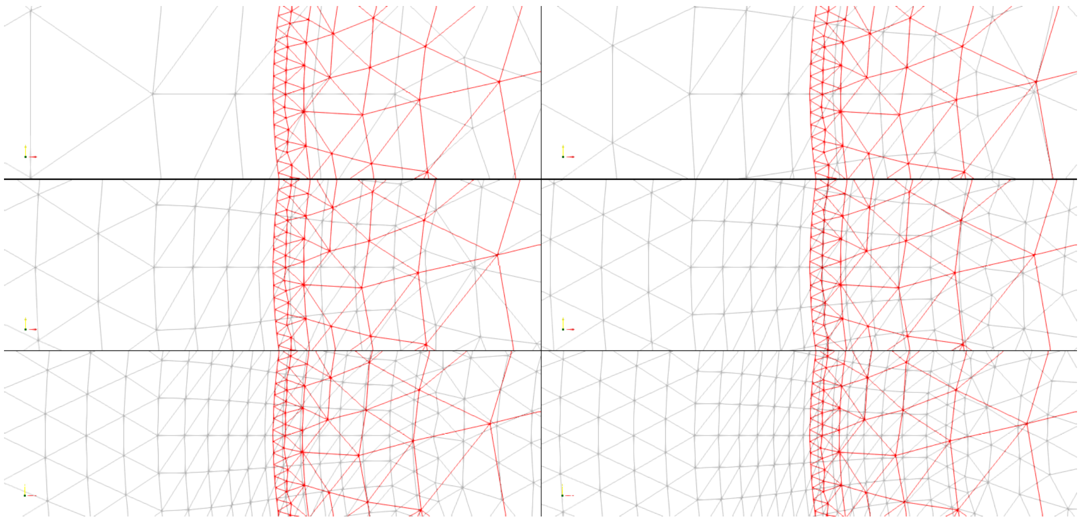


Fig. 4.12 Close-up of the meshes used for the 2D flow around a solid cylinder.

Mesh	Divisions	Δx	Δt	N triangles	N Nodes	N Δt
1	3	0.1333	0.1833	2588	1320	709
2	5	0.08	0.11	4820	2452	1182
3	7	0.0571	0.0786	9646	4881	1655
4	9	0.0444	0.0611	14294	7222	2127
5	11	0.0364	0.05	19130	9656	2600
6	13	0.0308	0.0423	24784	12499	3073

Table 4.2 Numerical data used in the 2D flow past a cylinder

Case	St
1	0.17582419
2	0.18807036
3	0.19467127
4	0.1965354
5	0.19690587
6	0.19708951

Table 4.3 Numerical results for the 3D flow past a cylinder

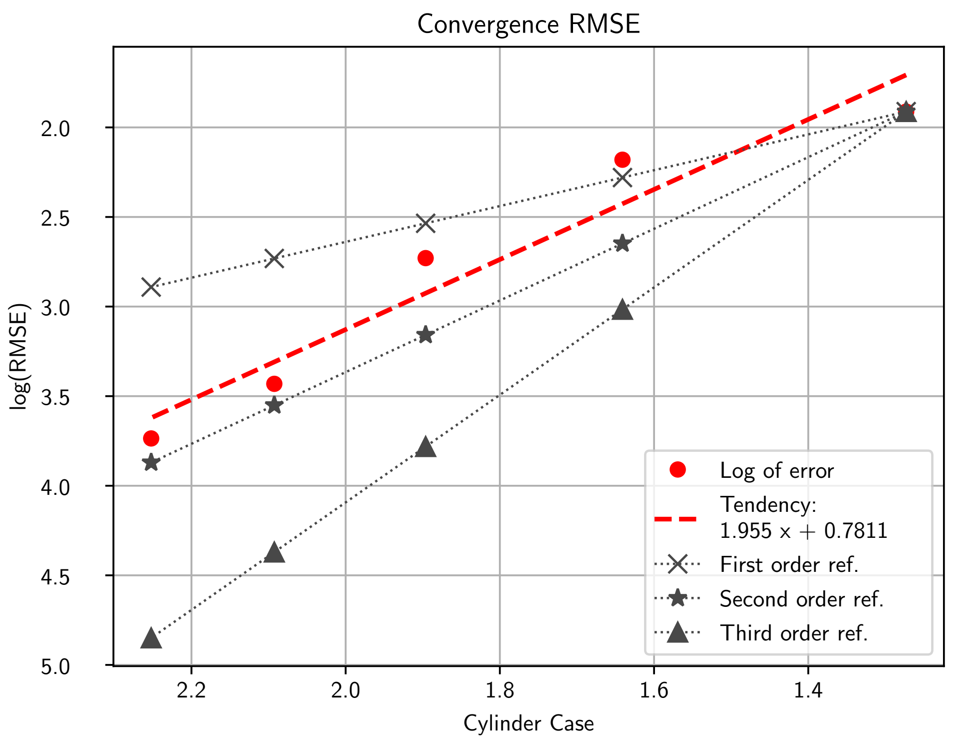


Fig. 4.13 Convergence rate of the 2D cylinder using solid intersections.

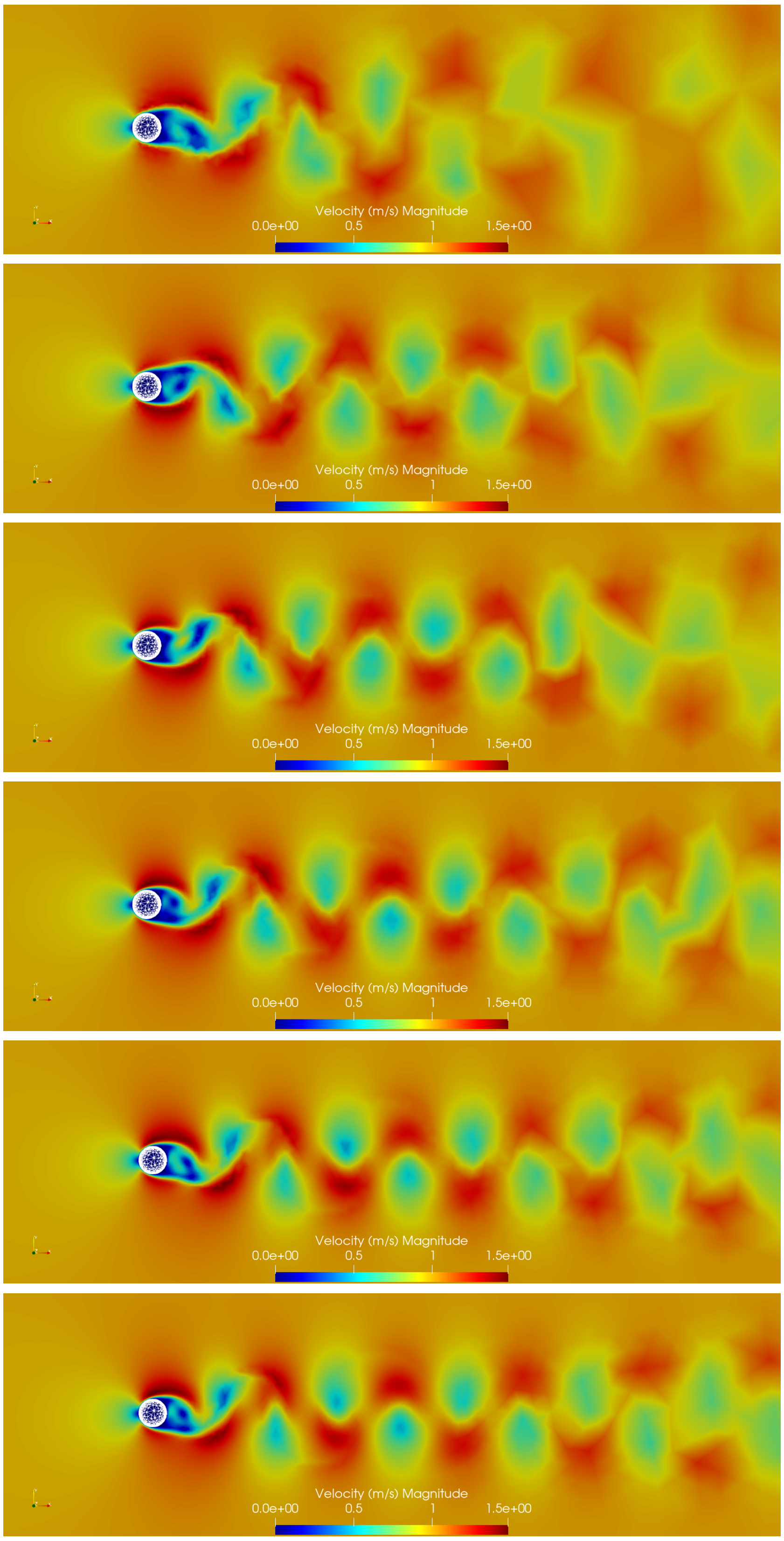


Fig. 4.14 Velocity profiles of the different cases of the 2D cylinder using solid intersections.

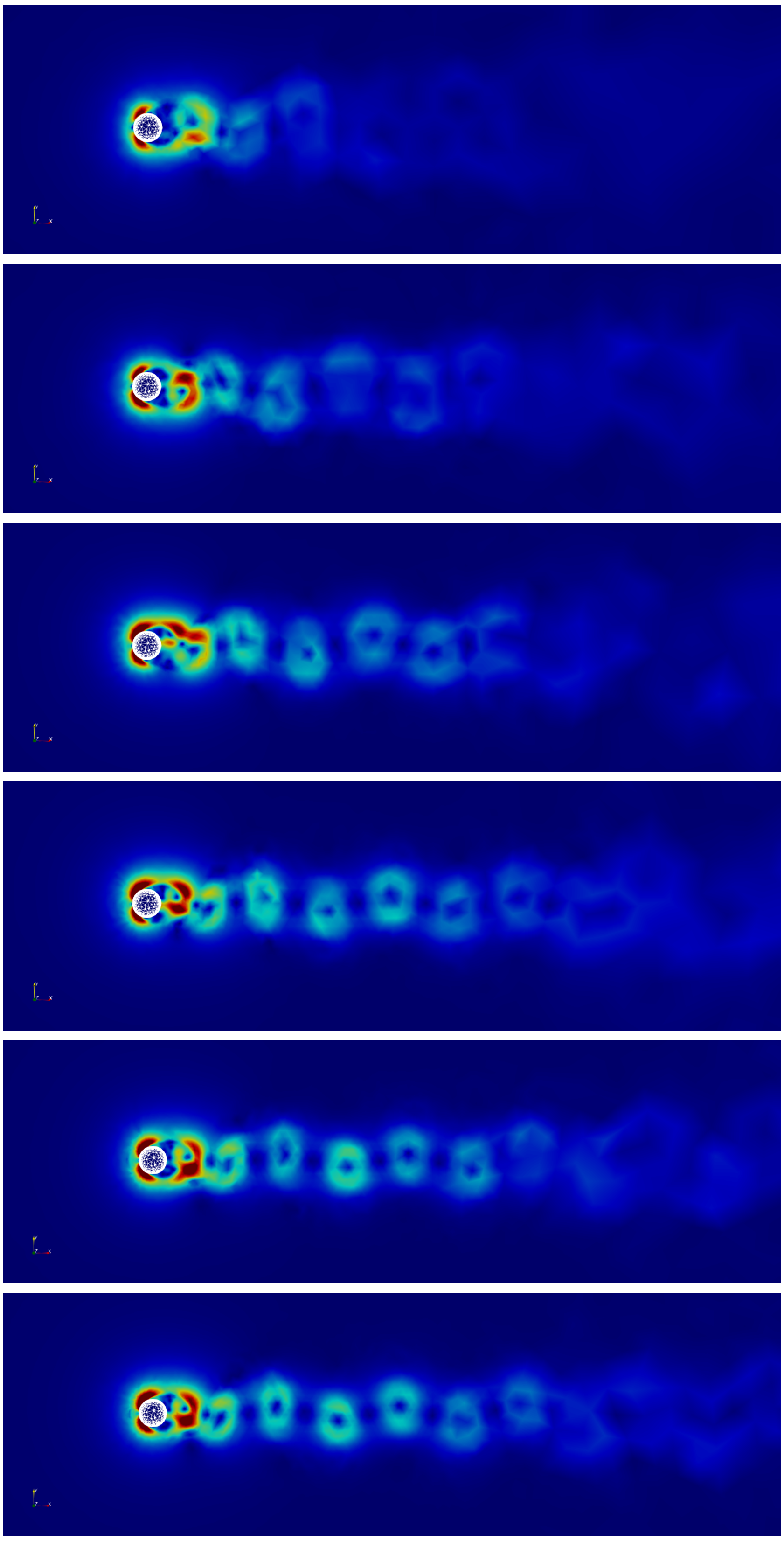


Fig. 4.15 Pressure gradient profiles of the different cases of the 2D cylinder using solid intersections.

4.4.2 2D flow around a cylinder - Strouhal comparison

This validation is a continuation of the previous validation. This time a full set of cases are run to obtain the strouhal numbers at each reynolds in order to compare against the different results shown at [JC17; RD53]. To perform the analysis, the sixth case of the previous validation is taken as a base.

Chapter 2, validation 6 2.6.7 performs the same validation but using a mesh object, not a solid object. Being the problem data the same and presented at table 4.4.

Cylinder diameter	$1\,m$
Inlet velocity	$1\,m/s$
Fluid Density	$1\,Kg/m^3$

Table 4.4 2D flow around a cylinder problem properties.

As with validation 2.6.7 the fluid characteristics are changed in order to obtain different reynolds numbers for each case. Table 4.5 presents the different cases where it can be seen that different viscosities are used to obtain the reynold number of each case allowing to keep other properties constant for the sake of simplicity.

Re	$\rho\,[Kg/m^3]$	$V\ [m/s]$	$R\ [m]$	$\nu\ [m^2/s]$
100	1.0	1.0	0.5	0.01
200	1.0	1.0	0.5	0.005
300	1.0	1.0	0.5	0.00333
400	1.0	1.0	0.5	0.0025
600	1.0	1.0	0.5	0.0016667
800	1.0	1.0	0.5	0.00125
1000	1.0	1.0	0.5	0.001

Table 4.5 Physical parameters used for the different cases.

Once the different cases are run, an FFT and a function fitting is performed on the lift forces in order to accurately obtain the vortex shedding frequency and the lift force amplitude. Figure 4.16 presents the different strouhal numbers obtained for different Reynold numbers and compared with the experimental curve presented at [JC17] and the same case with a mesh wall. As it can be seen good agreement is obtained.

4.4.3 2D heave oscillation of a floating cylinder

This validation is based on the experimental work carried out in [Itō77], where a circular cylinder is released from a defined height over a water tank allowing it to oscillate in

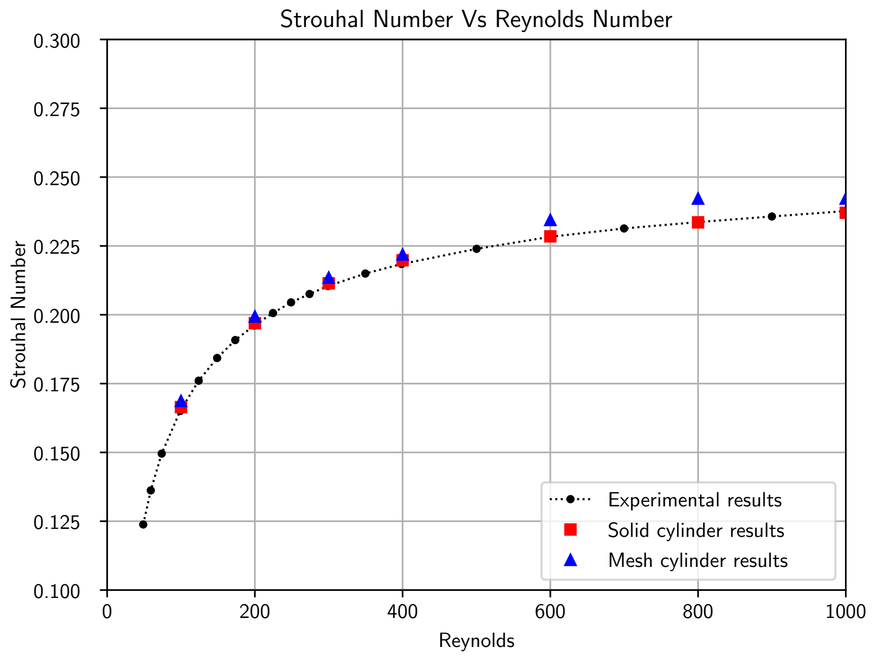


Fig. 4.16 2D flow around solid cylinder: Strouhal numbers for the different Reynolds numbers.

the heave direction. The result to compare here is the how the amplitude of the heave oscillation is gradually diminished over time.

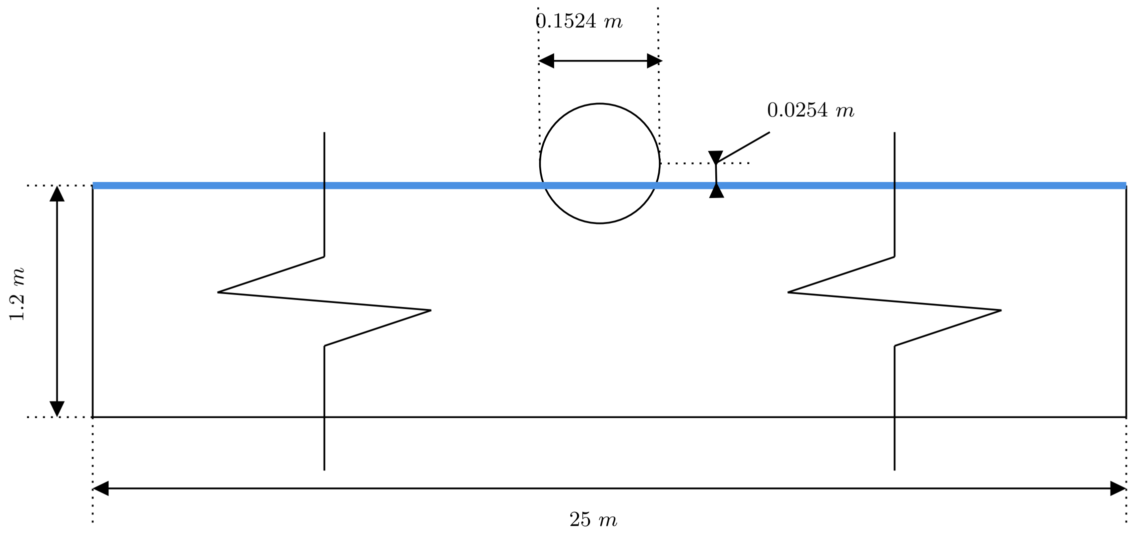


Fig. 4.17 Validation 3 problem description.

The problem data can be obtained from the first test of the reference [Itō77]. In this first test case, the cylinder of diameter $0.1524\,m\,(6\,inches)$ is located $0.0254\,m\,(1\,inch)$ above

the water-line.
The defined thank is $25\,m$ long, avoiding wave reflection of the walls, and the depth of water is of $1.2\,m\,(4\,ft)$. All of this can be properly observed from figure 4.17.

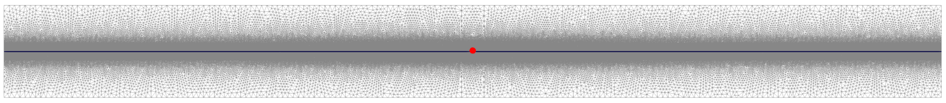


Fig. 4.18 Validation 3 generated mesh over the full domain.

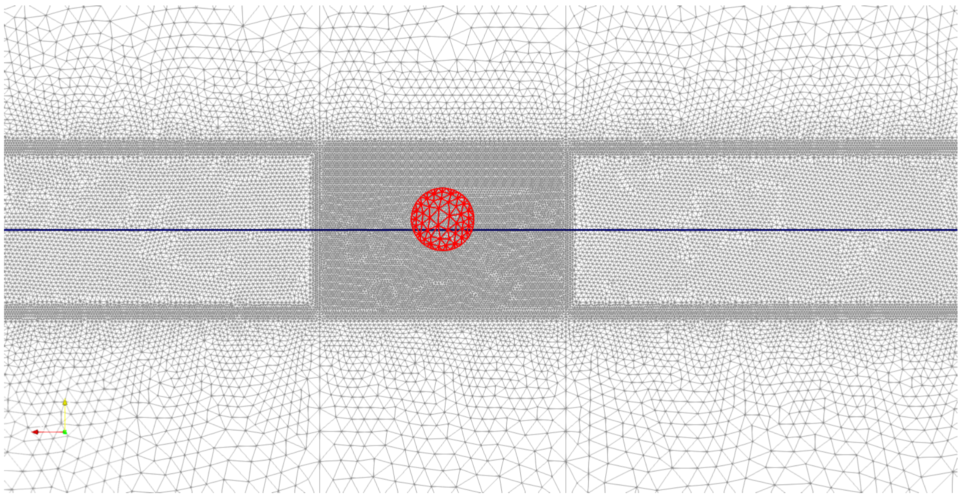


Fig. 4.19 Validation 3 mesh close-up to the cylinder (red) over the background mesh (blacj) and with the initial position of the free-surface(blue).

The generated mesh is refined close to the cylinder and the free-surface in order to properly capture the generated waves and the cylinder's oscillation. Figures 4.18 and 4.19 show the whole mesh over the problem domain and a close up to the cylinder. Both figures show the cylinder in red over the background mesh in black and the initial position of the free-surface in blue.

After computing a total of $2.5s$ of simulation time using a time-step of $\Delta t = 0.01$ the heave oscillation of the cylinder can be compared against the experimental results. These can be seen at figure 4.20 where good agreement of the vertical oscillation can be observed.

Finally, figure 4.21 shows the X-axis velocity contours close to the cylinder allowing us to observe the velocity profile that results from the generated waves of the cylinder's oscillation.

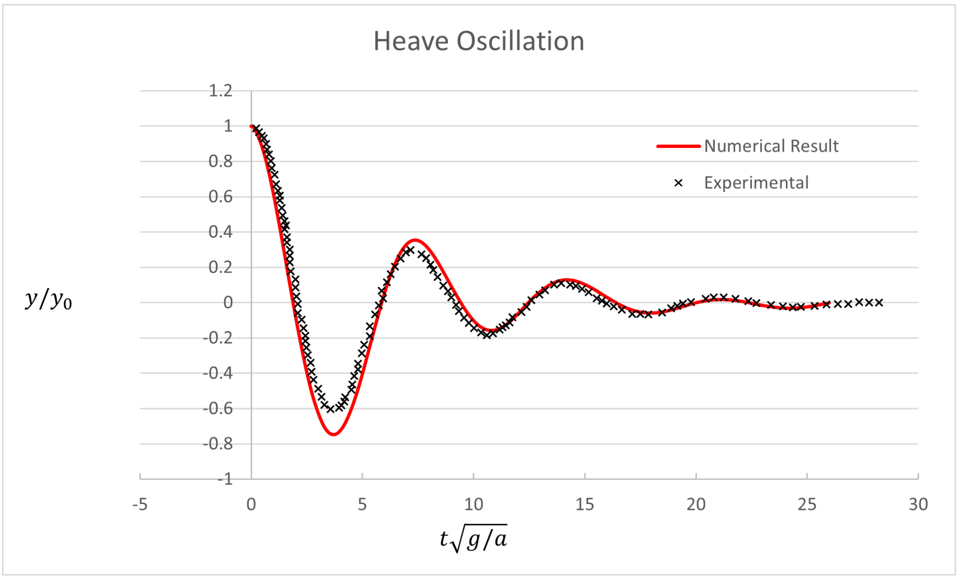


Fig. 4.20 Heave oscillation of the cylinder obtained experimentally from [Itō77] (black) and numerically (red).

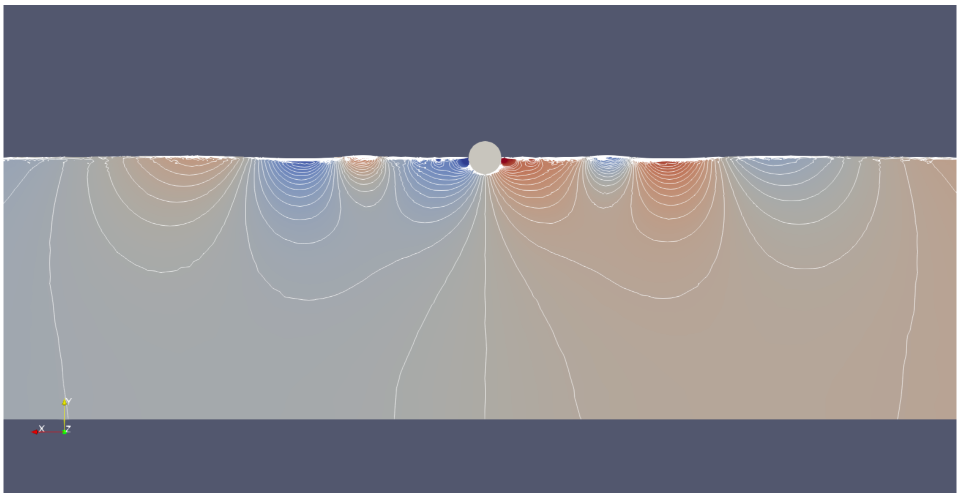


Fig. 4.21 X-Axis velocity profile obtained after 1.5s of simulation.

4.4.4 Show case: Patrol boat navigating through small waves

This last case is not a validation case but simply a show case of the capabilities of the tool. The case shows a patrol boat navigating through a set of waves with free heave and pitch degrees of freedom. The idea is to see the movements and force responses of the ship while crossing the waves.

The model, shown at figure 4.22, was taken from the popular website "GrabCad" and the link to the full model can be found at [Fur]. The particulars of the model are $LOA = 25\,m$,

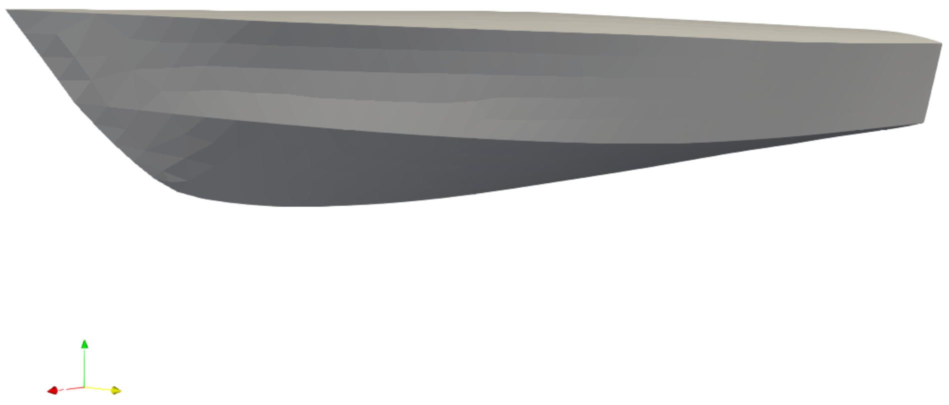


Fig. 4.22 3D View of the Patrol Boat.

$BOA = 7.1\,m$ and a displacement of $58.097\,tonnes$, the line plan is presented at figure 4.23.

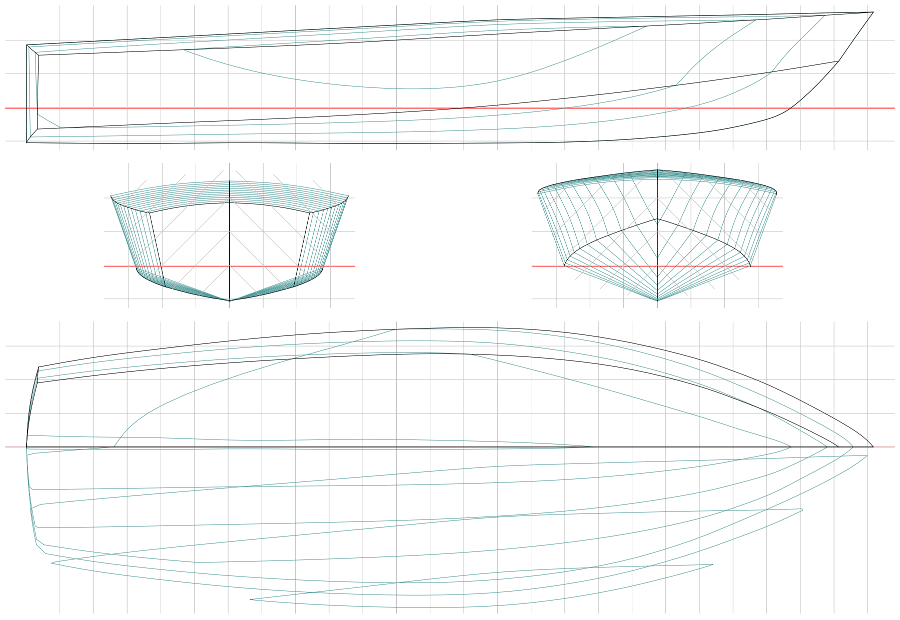


Fig. 4.23 Plan of lines of the Patrol Boat.

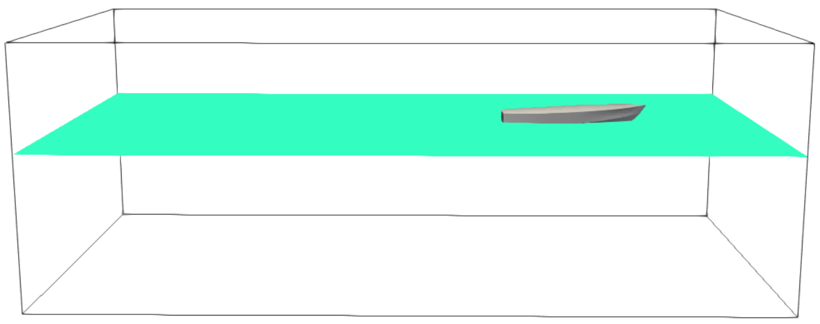


Fig. 4.24 Fluid domain created around the Patrol Boat.

Over this model, a fluid domain is constructed (shown at figure 4.24). It defines a $120\,m$ by $62.5\,m$ channel with a water depth of $25\,m$. The mesh is defined so that it is refined close to the patrol boat and to the free-surface. A close up of the mesh can be seen at 4.25. To simplify the process of generating the mesh, the domain is divided so that a structured mesh can be defined by simply applying a number of divisions. The rest of the domain is defined as an unstructured mesh with a large element size which allows to obtain a smooth transition from the zone of analysis. Figure 4.26 presents the small sub-domains and the chosen divisions. After generating the symmetric structured mesh, a total of 765.193 tetrahedral elements and 131.653 nodes are generated.

The boundary conditions applied are an inlet volume where a $16\,m/s$ velocity in the X-Axis direction is defined and where the waves are generated. This velocity is chosen in order to obtain a Froude number of about $Fr = 1.0$. The walls are defined using slip conditions and, finally, a pressure point of $P = 0\,MPa$ is defined at the upper corner of the outlet. The waves the patrol boat will be crossing is a monochromatic wave of amplitude $A = 0.2\,m$.

The simulation is run for 17 seconds and a time-step of 0.03 seconds is chosen. With this, the results obtained can be viewed in the following figures. Figure 4.27 presents the wave profiles generated once the patrol boat starts navigating through the inbound waves.

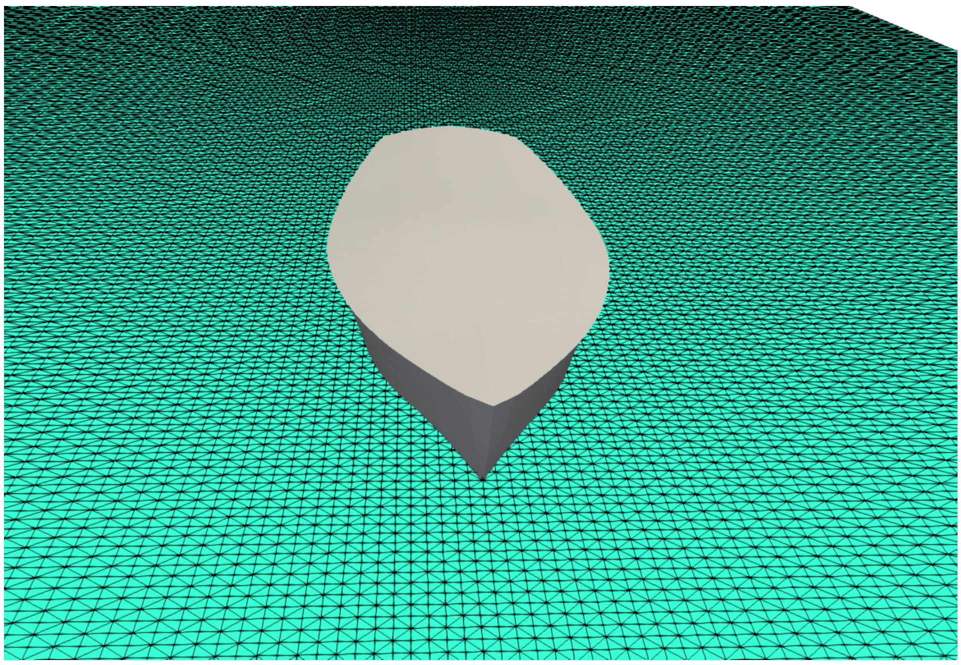


Fig. 4.25 Close up of the generated mesh at the Free-Surface and close to the Patrol Boat.

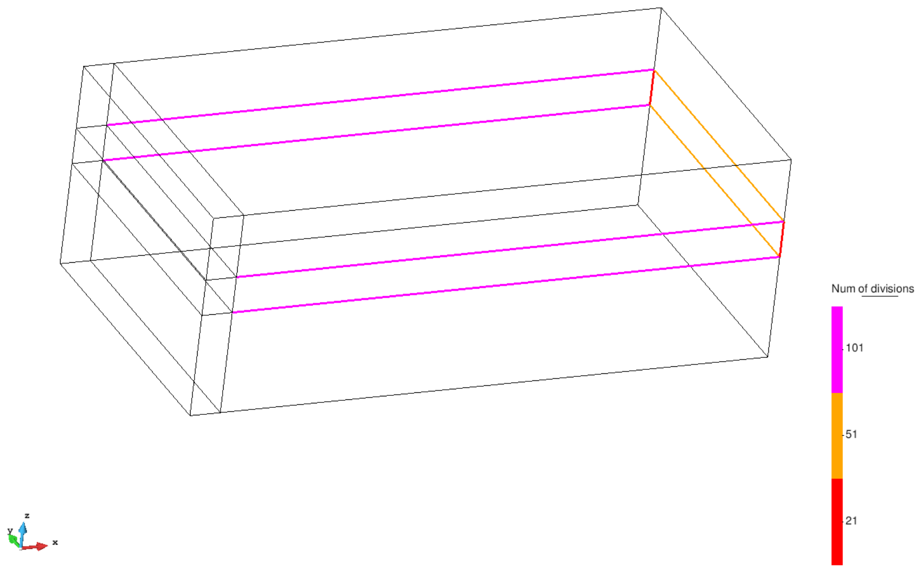


Fig. 4.26 Domain and created Sub-Domains with the chosen mesh divisions.

Figure 4.28 and 4.29 show the same image but figure 4.29 shows the free-surface defined by the particles and it shows how a much greater definition of the interface is obtained.

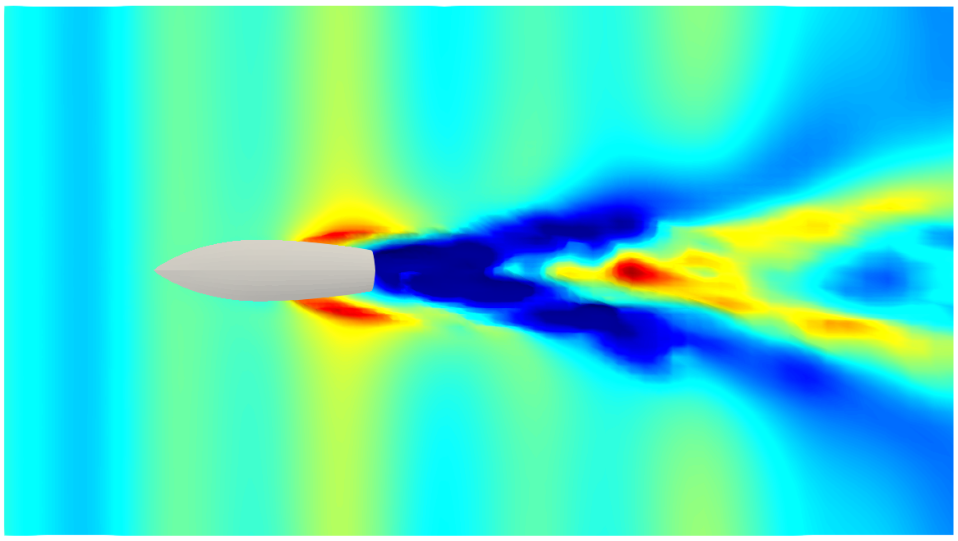


Fig. 4.27 Waves generated from the Patrol Boat while navigating through inbound waves.

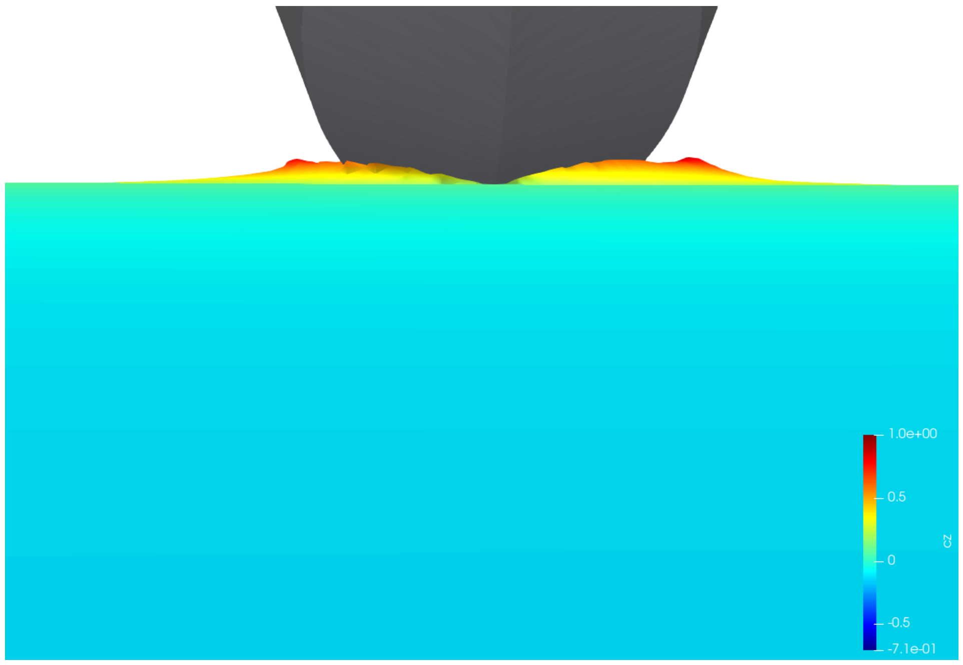


Fig. 4.28 Close up to the generated wave of the patrol boat defined by the mesh.

Finally the ship motions are presented at the two graphs shown at 4.30 and 4.31. It can be seen how the problem is initialised where the patrol boat increases its pitch angle

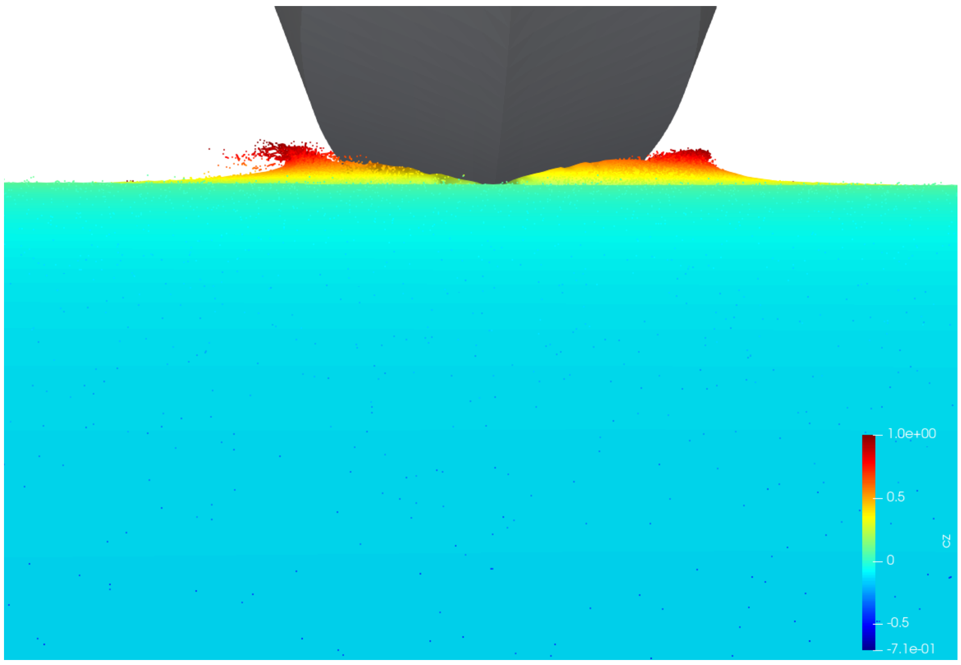


Fig. 4.29 Close up to the generated wave of the patrol boat defined by the particles.

and vertical position. This later changes as it starts navigating through the waves and a pitch and heave motion is observed.

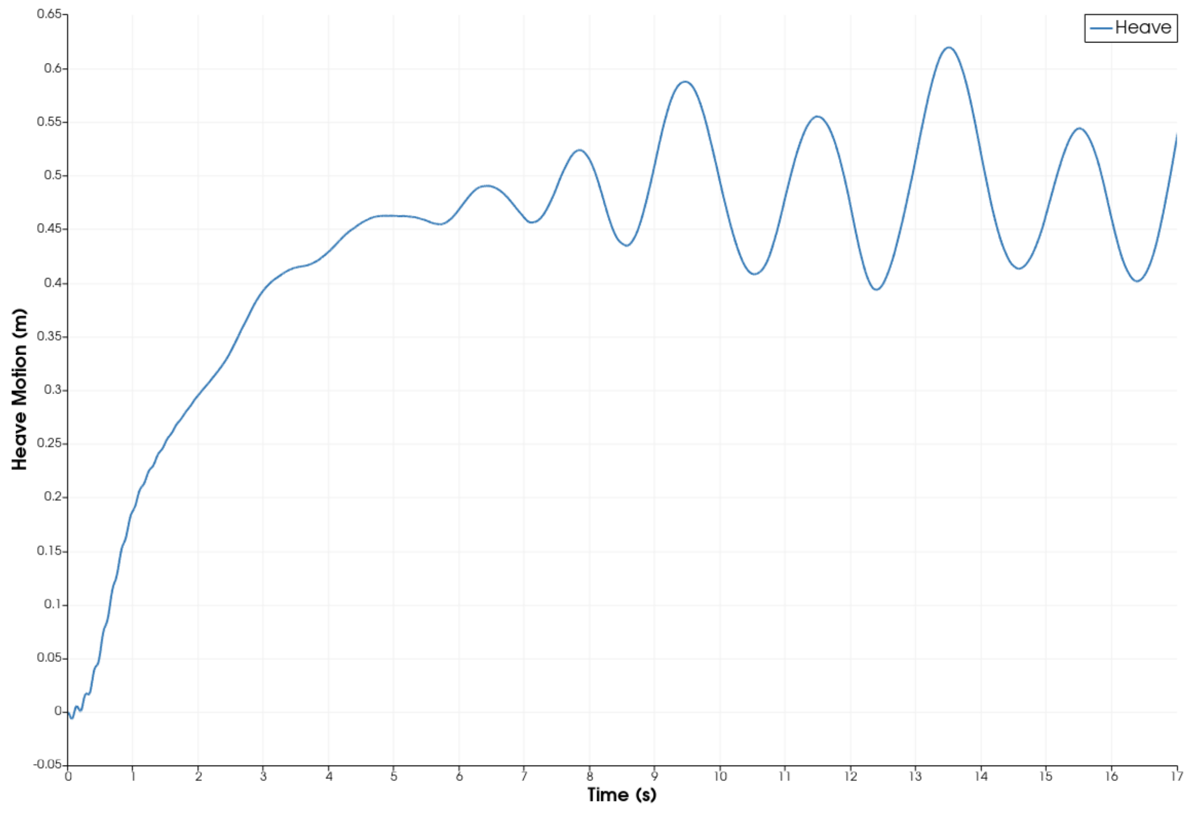


Fig. 4.30 Heave motion of the Patrol Boat over time.

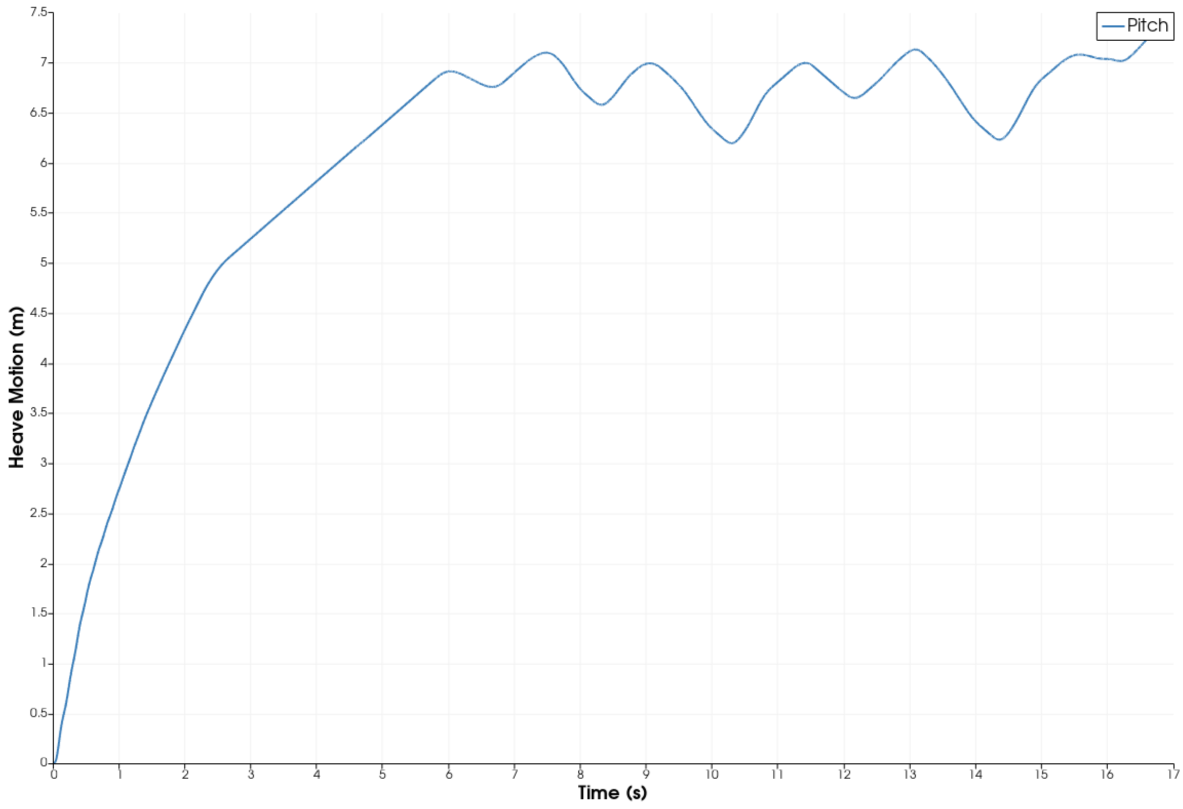


Fig. 4.31 Pitch motion of the Patrol Boat over time.

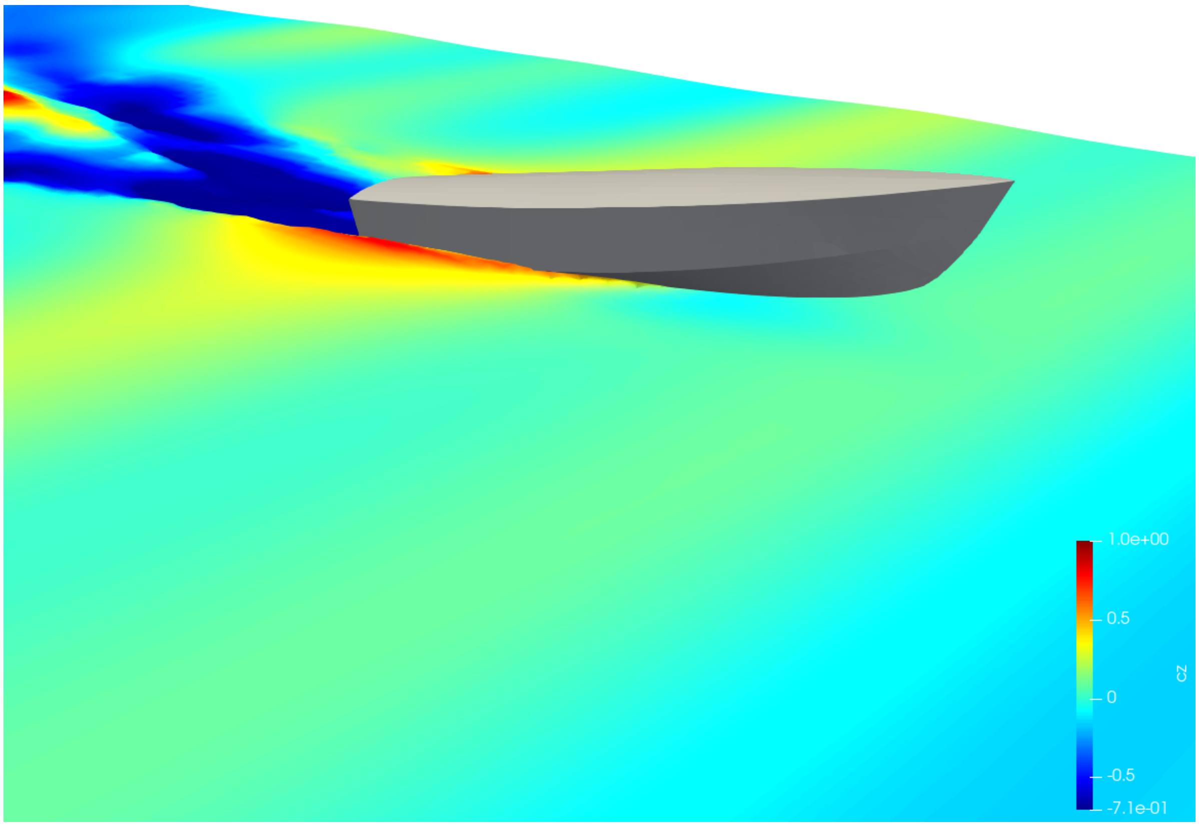


Fig. 4.32 Side View of the patrol boat while navigating through waves.

Chapter 5

Code Implementation and Optimisation

As the name of the chapter suggests, this chapter focuses on the development of the programming side of the thesis. The scheme developed throughout the thesis has been implemented using C++ and thought from the beginning from a parallel stand point.

This chapter will be presenting the different methodologies used to obtain a program that is capable of running on HPC computer clusters. To do this, domain decomposition will be used which partitions the problem domain into sub-domains capable of setting up the local algebraic system and provide the results of local operations done on them. This approach assumes that the global problem and all of the algebra involved to obtain the solution can be divided into a superposition of the local sub-domains. This superposition means that communication between sub-domains must be performed in order to obtain the global solution, therefore, in an HPC environment where memory is distributed among systems, these communications are done using message parsing protocols such as the ones found in MPI (Message Parsing Interface).

5.1 Code implementation

5.1.1 Basic concepts

The program developed is purely based on C++ and its standard library, and MPI is used to perform the parallelisation. This allows to use a parallel environment in any type of configuration: From a single computer to an HPC environment. This way the program is easily scaled to whatever needs a user can have.

Without getting into too much detail on computer hardware and its architecture, as it is not the goal of the thesis, the parallel strategies follow two main memory distributions: Shared memory and distributed memory. The first one is the one found in the typical day-to-day computer where the CPU-cores can access the same memory. The latter is found typically in HPC clusters where many single computers work simultaneously to produce large computational power. In such cases memory is distributed among the computers having to implement communication schemes that pass data throughout the computer cluster.

So two parallel strategies can be used: one that benefits from using the shared memory configuration and another from the distributed memory. If a shared memory implementation is chosen, such as OpenMP or the threading library found in Boost and the standard library of c++, then it cannot be used in a distributed memory environment. But if a distributed memory implementation is chosen using message parsing protocols, then it can be used in any of the desired cases as it can be used in shared memory environments where it simulates a distributed memory scenario. In fact, a combination of the two is also possible where each individual machine in the cluster is within the distributed memory environment and use communications schemes between one-other and within each machine a shared memory environment is used.

This thesis chose the distributed memory environment that allowed to scale the program in order to make use of computer clusters such as the one found in CIMNE. Therefore the known message parsing interface (MPI) is used for communication between the different computers.

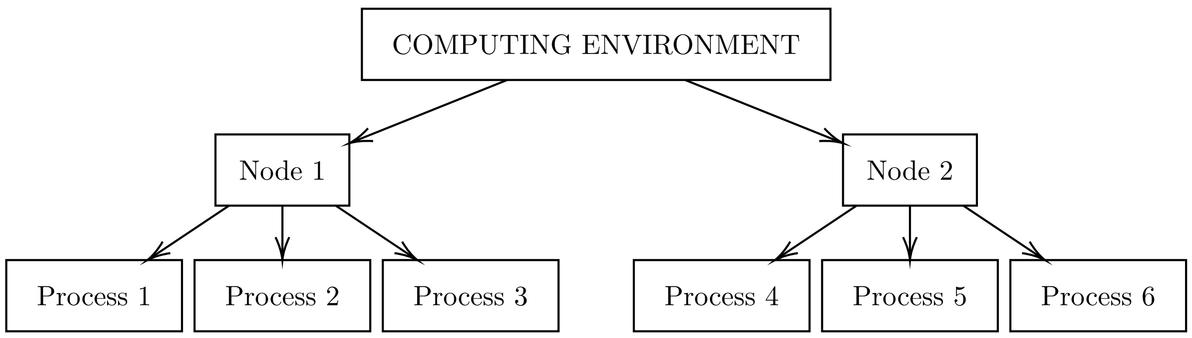


Fig. 5.1 Node and process distribution

In a distributed memory enviornment using MPI, two main definitions appear: Processes and Nodes. A process represents one processing unit, having as many as one might need. All of these processes are grouped within a number of nodes. As an example figure 5.1 shows a configuration in which each node contains a total of 3 processes. Without getting into much detail on how the processes are ordered and allocated throughout the nodes, in this thesis a process to node ratio is chosen which balances the operations throughout the compute nodes. For example, if three nodes are available and each one can handle 12 processes independently, and we launch a total of 27 processes then the processes are distributed among the nodes in such a way that they are balanced: 9 processes per node.

To further understand how this is viewed from a hardware stand point, a node can be viewed as one computing machine which contains a number of processors and each processors has a number of physical cores which can perform operations independently. For example, one node contains 2 processors and each processor contains a total of 24 physical cores, then this particular node configuration allows to use up to 48 processes which will run independently from one-other. More than 48 processes could be used in this case, but they would not be able to run at the same time, finding that processes would pause while others are performing operations.

5.1.2 Domain Decomposition

When creating a program that needs to make use of distributed memory parallel strategies two well defined paths can be taken: a global algebraic system divided by rows or columns throughout the different processes or divide the problem domain into as many smaller sub-domains as available processes having a one-to-one ratio of these new smaller sub-domains and processes (one per compute process).

The first scenario tends to be implemented when the application in hand can be used by a large amount of other applications. Such is the case of solver technologies, where a lot of numerical schemes can make use of them and each of them has different parallel implementations. A solver library cannot take into account all of the scenarios so a general global parallel scheme is chosen.

The main problem with this type of strategy is that when using mesh based algorithms such as FEM, the assembly of algebraic systems and operations are more dependant on the elements and not on the node numbers. Therefore, if the problem is divided into chunks of global nodes then elements are not necessarily assembled within each process overcomplicating the parallel implementation and data structures. For example,

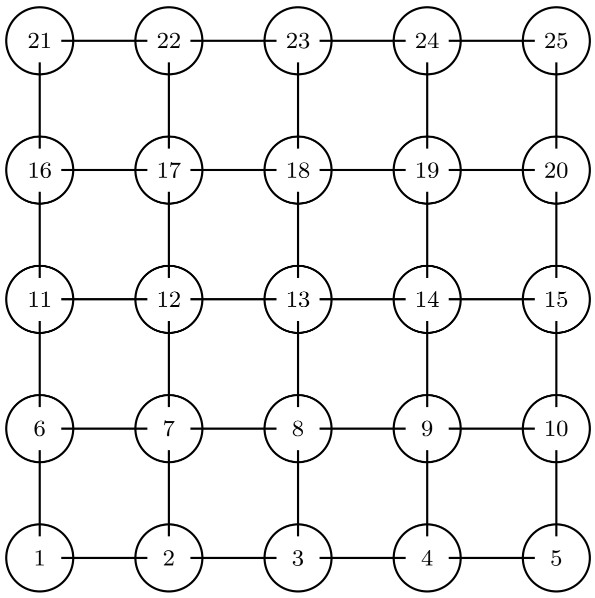


Fig. 5.2 2D 4x4 square domain

figure 5.2 presents a simple 2D 4x4 square domain divided into 4 node elements. If the nodal enumeration is used to divide into chunks the global system of equations, then we would have a divided mesh as shown at figure 5.3. Though existing the possibility of implementing a parallel structure, it is not the most obvious one and can overcomplicate the problem.

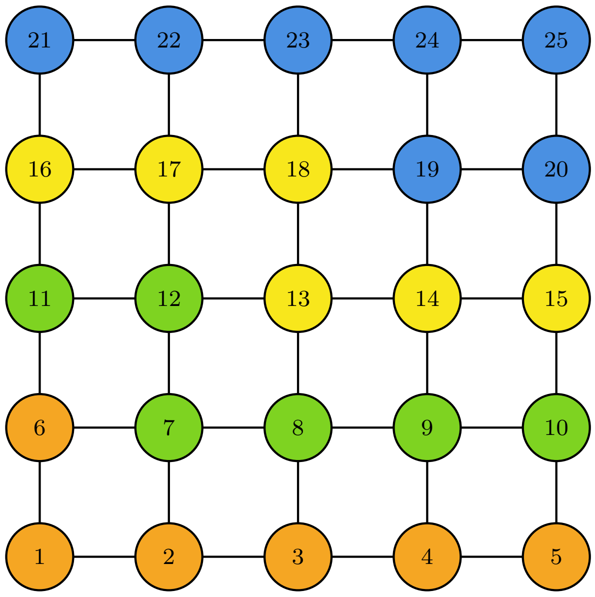


Fig. 5.3 Global node distribution for the parallel strategy

Let's now consider the particles that we have in the scheme presented in this thesis. As the particles belong to elements and these particles move between elements, we would have to construct another parallel environment just for the particles as the problem is divided nodally and not by elements.

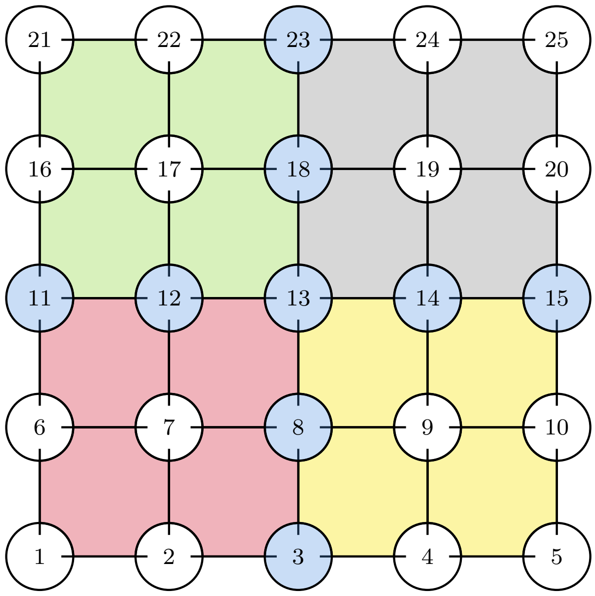


Fig. 5.4 Sub-Domains generated from the initial $4x4$ domain

A more obvious way of assembling the problem that also takes into account the particle distribution would be by dividing the problem domain into local sub-domains that contain a balanced amount of elements. This allows for local assembly of the system of equations and a later global construction where only the shared-nodes have to add their values. Figure 5.4 shows this scenario where the domain presented at figure 5.2 is now divided into the coloured sub-domains.

Furthermore, by dividing the domain into local sub-domains by elements, particle operations become embarrassingly parallel. This is because particles will be moving within the local sub-domain without needing information of other sub-domains unless a particle moves from one sub-domain to the other.

As this domain partition is the appropriate approach for the SL-PFEM scheme, a known library is used called METIS [Kar20]. This library has been thoroughly worked on and has the ability to efficiently partition very large domains into sub-domains by minimising the shared nodes and obtaining sub-domains that have a balanced number of elements. This balance of elements allows to properly spread out the domain's elements without having processes that have to calculate larger local systems than others.

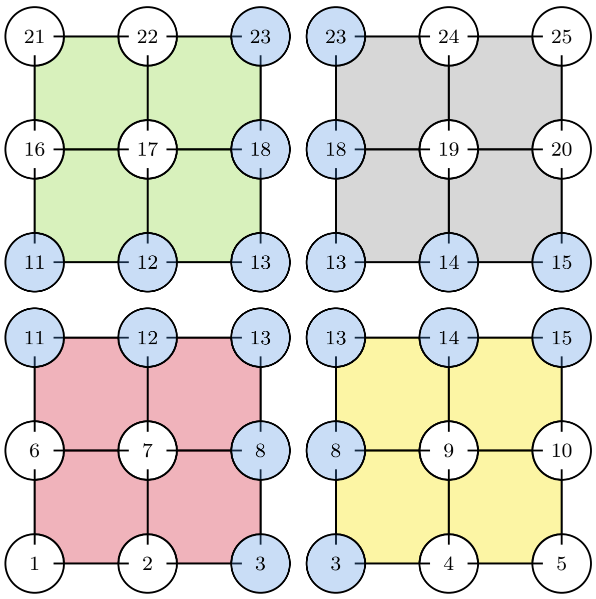


Fig. 5.5 Shared nodes of the generated sub-domains.

5.1.3 Basic Data Structures

Once the basic implementation features of the program have been shown, an overview of the main basic data structures are presented. These structures represent the core of the program as other routines and classes will depend on them.

To avoid data repetitiveness, structures are built around core data-types. For example, on the mesh side, the lowest level data-type in the current program would be the nodes, which contain the nodal coordinates and nodal values. With these nodes, elements can be constructed allowing to access nodal information from an elemental structure.

This methodology, even though it might take an initial time overhead to build, allows to control the amount of memory being used and prevent data duplicity. Figure 5.6 shows the different data structures and how one depends on the other.

Figure 5.6 shows the hierarchy of the structures used on the mesh side, where the lowest level are the nodes. Then elements that are built over said nodes, then the sub-domains which contain nodes and elements and, finally, the domain class which contains all of the sub-domains. By building this hierarchy, all of the information that is below one level is available at all times.

The same reasoning is applied to the particles. In this case, particles contain their position and values that will be transported. But in this case instead of being fixed values as the nodes, they are able to move around freely, ultimately crossing sub-domains or

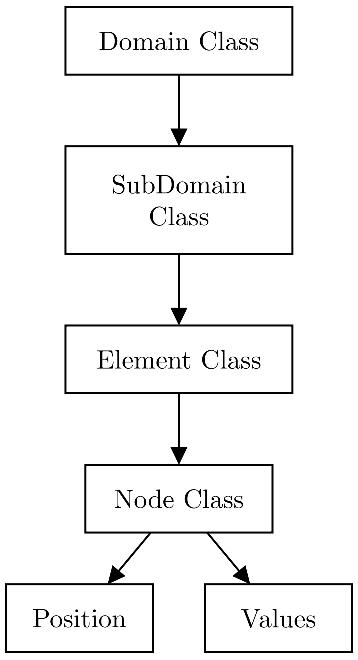


Fig. 5.6 Hierarchy of data structures.

even leaving the problem domain. This means that particles need an extra data structure that allows to manipulate particles that move from one sub-domain to another.

Figure 5.7 shows a general scheme of the different basic data structures and how the different parts of the SL-PFEM scheme depend on them.

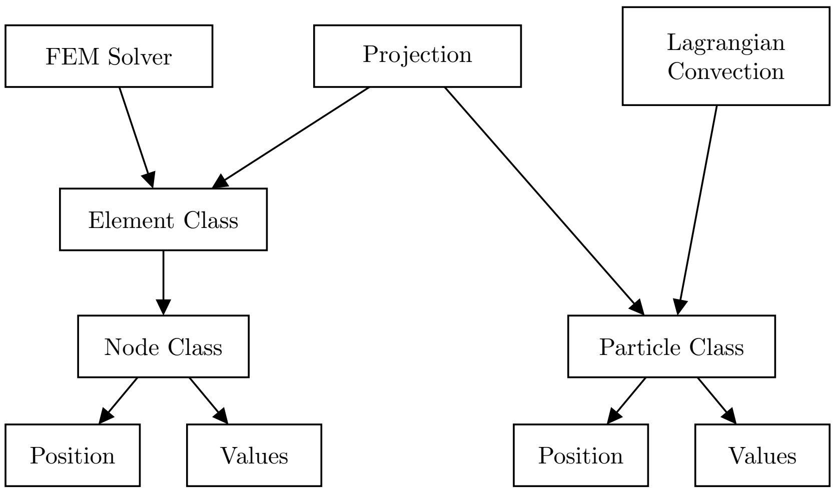


Fig. 5.7 Dependencies of the different data structures.

5.1.4 Parallel particularities

Up to now the general basic data structures have been presented but when applying them to a parallel environment extra information and data structures are needed in order to manipulate and obtain global results.

To analyse the mesh parallel strategy, we take the lower left sub-domain of figure 5.5. By using sub-domains, the advantage is that local assemblies can be done at each sub-domain. Therefore, a local enumeration is needed that allows to easily assemble the data into local matrices. Figure 5.8 shows how the chosen sub-domain is re-enumerated.

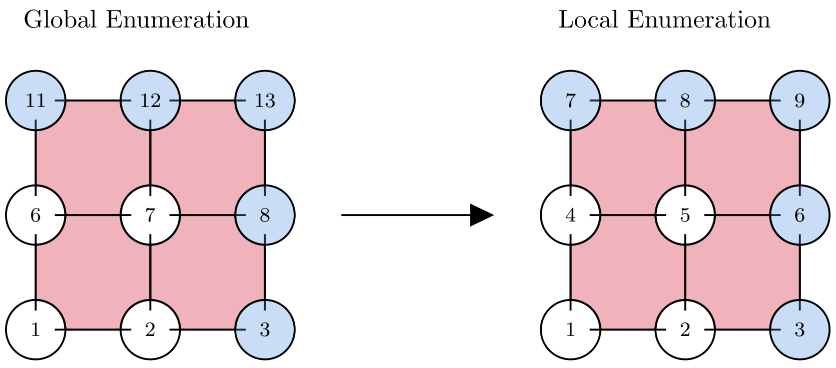


Fig. 5.8 Enumeration change between the global domain and the local sub-domain.

So, to do this, the necessary information is to have a local index to global index for the nodes that allows to find the global enumeration of the locally enumerated nodes. Furthermore, another data variable that will be necessary is to find which nodes are shared and to which processes they belong to. This will allow to easily find and share data between the different processes.

For the particles a different approach is taken. The difference is that particles are moving throughout the domain while the nodes are fixed in space. Therefore two sets of particles are available. One is the full list of local particles that are found within the sub-domain. These particles are transported and when a particle passes to another sub-domain then it is inserted into the other list of particles. This second list acts as a buffer to move particles from sub-domain to sub-domain (process to process). It will generally be empty and will only be filled by crossing particles. Once the particles have been passed to the new sub-domain then the buffer list is emptied. This process is repeated at each time-step. Algorithm 4 showcases these steps.

Algorithm 4: Parallel strategy for the particles using the two lists of particles.

```
while (Time Loop: 1 to n $\Delta t$)
{
    Move local list of particles within sub-domain
    If a particle has moved to other sub-domain insert it into the buffer
    while (Buffer Particles != empty)
    {
        Check if particles from the buffer belong to the new sub-domain
        if (Buffer Particle belongs to sub-domain)
        {
            Delete the particle from buffer and add to local sub-domain list of particles
        }
        else
        {
            Insert the particle into the buffer again and send to other sub-domain
        }
    }
}
```

5.2 Scalabilty

Now that the basic structure of the program has been presented, it must be put to test to see that it behaves as expected. In order to do this, two scalability tests will be analysed: Strong scalability and weak scalability.

Scalability is a widely used variable that defines how hardware and software can deliver more computational power when the resources are increased, e.g. with more resources a program can run faster. When analysing the performance of software, then scalability is referred to as parallel efficiency, this is the ratio between the obtained speed-up and the ideal speed-up when using a certain amount of processes. In this case, speed-up is defined as:

$$Speed - up = \frac{t_1}{t_N} \tag{5.1}$$

Where t_1 is the amount of time spent running the program using one process and t_N is the amount of time spent running the same program over N processes. An ideal scenario would obtain a linear speed-up where the ratio is in fact equal to N meaning that each process would be using all of its computational power.

5.2.1 Strong Scaling

The concept of strong scaling was initially defined at [Amd67]. Here the time the program takes to finish is divided into two parts: One part is the serial part (s) which will never scale when using more than one process and the other part is the parallel part (p) which scales perfectly when using different amounts of processes (N). In fact, [Amd67] defines that the actual speed-up will be limited by the serial part of the program:

$$Speed - up = \frac{1}{s + \frac{p}{N}} \tag{5.2}$$

Therefore to properly view the strong scaling of a program, we must take a fixed size problem and use a different amount of processes to see how the speed-up for the different amount of processes varies with respect to an ideal solution. What we will see is that the problem will stop scaling as the serial fraction of the program overweights the parallel part.

This scalability variable is the most common variable used as it is the most intuitive one. It is the typical variable we would use to see the performance of a program on a day-to-day computer.

For example, let's take the 2D flow around a cylinder case from chapter 2 2.6.7 and generate a mesh that allows to run the case in one node of the CIMNE cluster that contains two processors with 10 cores each ($HighParallelization$ nodes), totalling a maximum of 20 concurrent processes. This way, we will have a total of 20 different times obtained from running this particular case with 20 different process counts (from 1 to 20).

The cylinder case is discretised using a total of 137885 nodes and 576000 triangular elements and a total of 5 particles per element are used. The time step is chosen that obtains a CFL of 2.0 close to the cylinder wall, and running a total of 500 steps. Results are not written as this benchmark is not intended to view the write efficiency.

With this, figure 5.9 shows the speed-up obtained for each case, being the dashed line the fitted curve based on [Amd67]. Furthermore a decrease in the speed-up can be seen when using more than 10 processes. This is due to external reasons as it is too much of a coincidence that when using 11 processes, meaning that we have exceeded the cores that one processor has by 1, then the speed-up decreases. This could possibly be by a miss-configuration of the distribution of tasks over the node when using the "slurm"

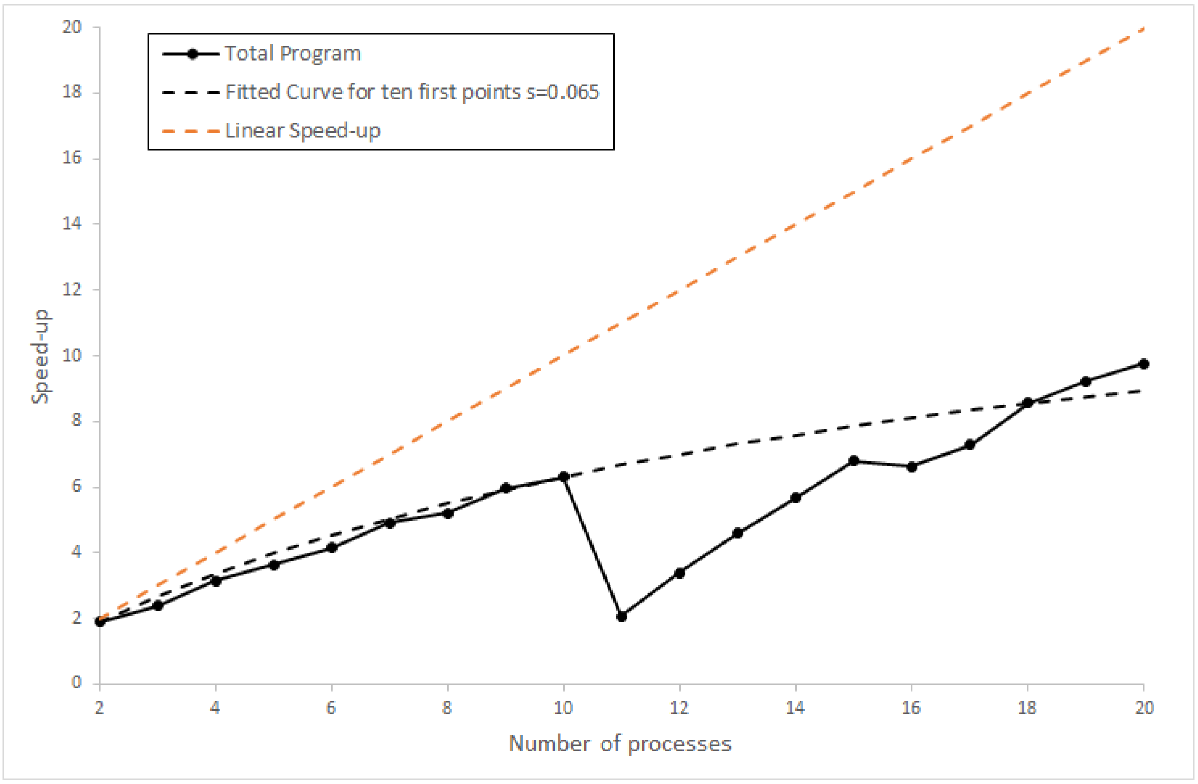


Fig. 5.9 Program Strong Scalability compared with ideal scalability. Dashed line is the fitted curve following [Amd67].

environment. So, the fitted curve based on [Amd67] is calculated based on the ten first points; obtaining that the serial part of the program is around a 6.5%.

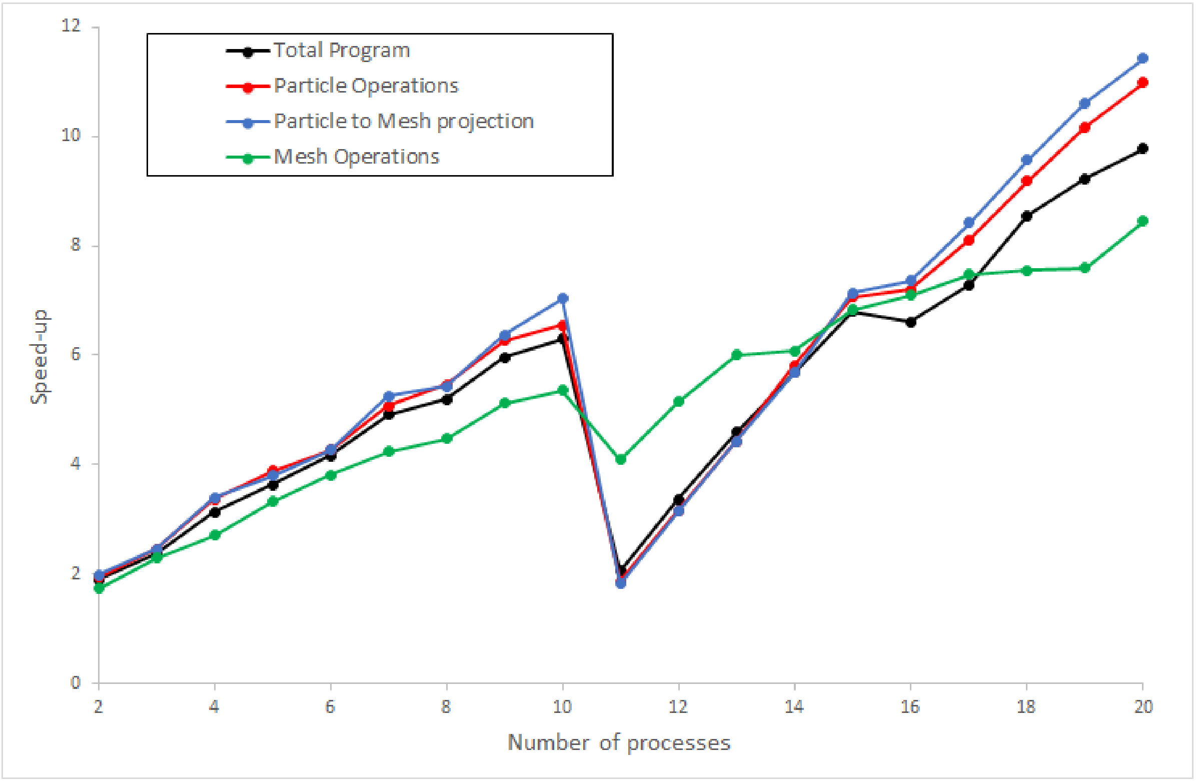


Fig. 5.10 Strong Scalability of the parts of the SL-PFEM.

Figure 5.10 shows a detailed graph of the strong scaling of the different parts of the code, differencing the mesh computations from the particle computations. In fact, a further separation is done on the particle side as the projector is influenced by the mesh. By taking this division, we can observe that, in fact, the particles scale much better than the mesh (solvers) concluding that the limiting part, when choosing the most adequate amount of processes, are the amount of nodes. So a ratio of nodes per process can be a good choice for an initial estimate of the amount of processes a problem can use.

Finally, a general way of choosing the maximum speed-up that is beneficial is when the obtained speed-up is half of the ideal speed-up. Therefore, if the ideal speed-up is 20 times, then the lowest speed-up that would be considered adequate would be 10 times. Concluding that when observing figure 5.10 and taking into account that the scalability is limited by the mesh side, then using a maximum of 10 cores for the problem in hand would be the maximum most efficient choice to obtain a reasonable speed-up for the current cluster ($HighParallelization$). This means that a maximum of about 10.000

nodes per process would be the minimum amount of nodes a process can efficiently handle.

5.2.2 Weak Scaling

The previous variable indicates the ability of scaling one problem by increasing the available hardware resources. In fact, it shows the upper-limit of the speed-up for a fixed problem size. This does seem to prove that a bottleneck exists for parallel computing because if one wanted to obtain a 100 times speed-up on 200 processes, then by using equation 5.2 the serial proportion of the program cannot exceed 0.5%.

As pointed out at [Gus88], in practice the sizes of the problems are increased with the amount of available resources. In other words, if a problem of a certain size only requires of a certain amount of resources, using more than these resources will not be beneficial. Therefore a reasonable approach is by using small amounts of resources for small problems and large amounts of them for larger problems.

With this, [Gus88] proposed a scaled speed-up formula where the parallel part (p) scales linearly with the amount of processes (N) and the serial part (s) of the program does not increase with respect to the size of the problem:

$$ScaledSpeed - up = s + p \times N \tag{5.3}$$

This defines a new variable called weak scaling where the scaled speed-up is computed based on the scaled problem size. This is different to strong scaling where the speed-up was computed for a fixed sized problem which is limited by the amounts of resources one problem can use.

To better understand this, the idea of weak scaling is to see if the program scales correctly when using large amounts of resources. For example a problem of size 1000 running on 10 processes should ideally take the same amount of time as using the same problem but of size $1e^6$ running on $1e^4$ processes.

This variable is generally analysed using large amounts of processes (in the order of thousands) such as the results shown by the used solver "HYPRE" [Bak+11; LLN21] where the weak scaling analysis makes use of a total of 200K processes. This analysis proves to be impossible to perform in this thesis as we do not have the possibility of using 200K processes.

Cylinder Case	Number of Nodes	Number of Elements	Time-Step	Cores used	Nodes per Core
1	224.991	1.265.472	0.0385	16	14.062
2	495.635	2.829.888	0.0294	32	15.489
3	688.389	3.950.784	0.0263	48	14.341
4	925.575	5.334.336	0.0238	64	14.462
5	1.211.801	7.008.192	0.0217	80	15.148
6	1.374.744	7.962.624	0.0208	96	14.321
7	1.743.170	10.123.776	0.0192	112	15.564
8	1.949.805	11.337.408	0.0185	128	15.233

Table 5.1 Weak Scalability cylinder cases.

The main point of focus for this analysis is to check the particle implementation as it is the most innovative point of the thesis and does not depend on external libraries such as the solver. Therefore, a small weak scaling analysis will be performed to check the how the program behaves.

Reference [Bak+11] shows that when few processes are used, poor weak scaling is found for the solver library "HYPRE". Taking into account that the analysis to be done here will be using a small amount of processes, then poor weak scaling of the solvers are expected.

To benchmark the developed program using a weak scaling analysis a similar analysis as the strong scaling is performed. The 2D flow around a cylinder case from chapter 2 section 2.6.7 is taken, and a variety of problems are generated, that change the problem size. Once this is done, they are computed at the CIMNE cluster using the $b510$ nodes where each node has two processors of 8 cores each.

Table 5.1 shows the problem sizes used (amount of nodes and elements) and the amount of processes used, where the amount of processes is chosen so that the ratio of nodes per process is fixed throughout the analysis. As it can be seen, the analysis starts using 16 cores, which is a full node. The idea is to use full nodes

In order to ensure that the same case is computed at each problem size, the time-step is reduced with the mesh-size in order to keep the CFL number constant to 2.0. Also, the same amount of steps are used, therefore each computation will have to perform the exact same number of operations. This means that the largest problem using the largest amount of resources will have each process giving out the same compute power as when only using one process for the smallest problem.

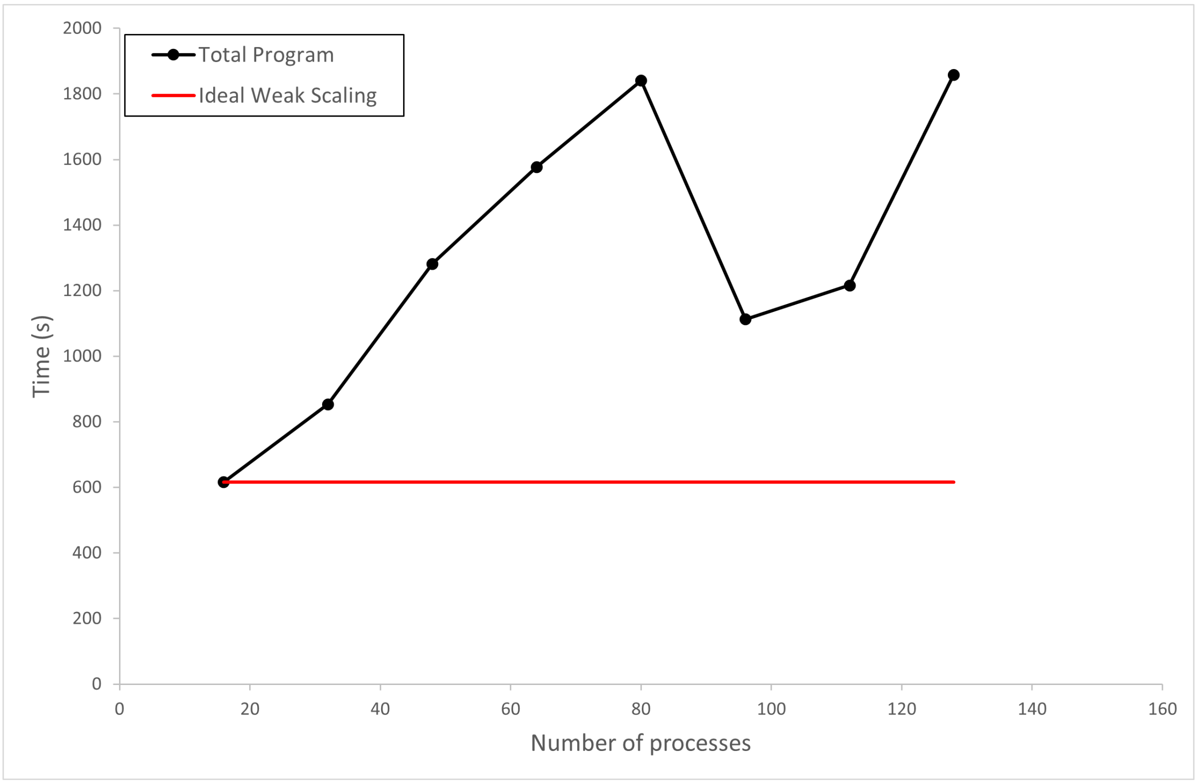


Fig. 5.11 Weak Scalability total problem times (black) compared to expected ideal times (red).

Figure 5.11 shows the times obtained, and it seems to show a relatively good week scaling compared to the expected time used for one node (16 processes), for example, using 128 processes (a problem about 100 times larger) it only takes 3 times more. Furthermore, the times decrease for the problems using 80 and 96 processes, which seems to be a problem with the compute node's usage.

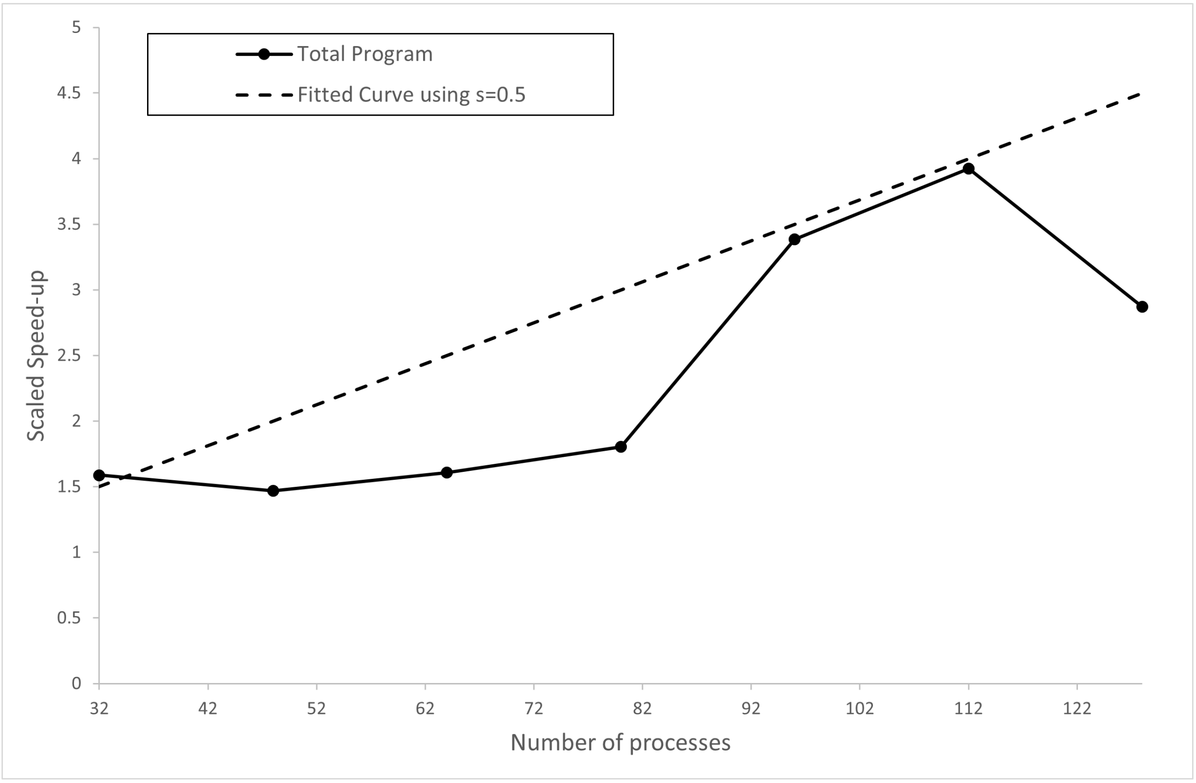


Fig. 5.12 Weak Scalability scaled speed-up compared to expected ideal scaled speed-up (dashed).

But figure 5.12 shows the scaled speed-up and here we can observe how the total weak scaling of the problem is not performing well. The dashed line shows the scaled speed-up using [Gus88] where we see that it is obtained for a serial fraction of the program of 50% which is not very impressive.

But, as it was explained before, the solvers can cause a problem as they do not show the good weak scaling until a large amount of processes is used. So, if we now plot the weak scaling for the different parts of the problem, then we can see the weak scaling for the solvers and for the particle operations. Figure 5.13 shows this division and allows us to observe how the solvers do not have good weak scaling but the particles do.

If we take use the formula proposed before to plot the ideal scaled speed-up against the particle scaled speed-up, then a much better result is observed. Figure 5.13 shows that the serial fraction for the largest node counts is below 20%. This is a good value that is expected for a program that runs at an HPC environment.

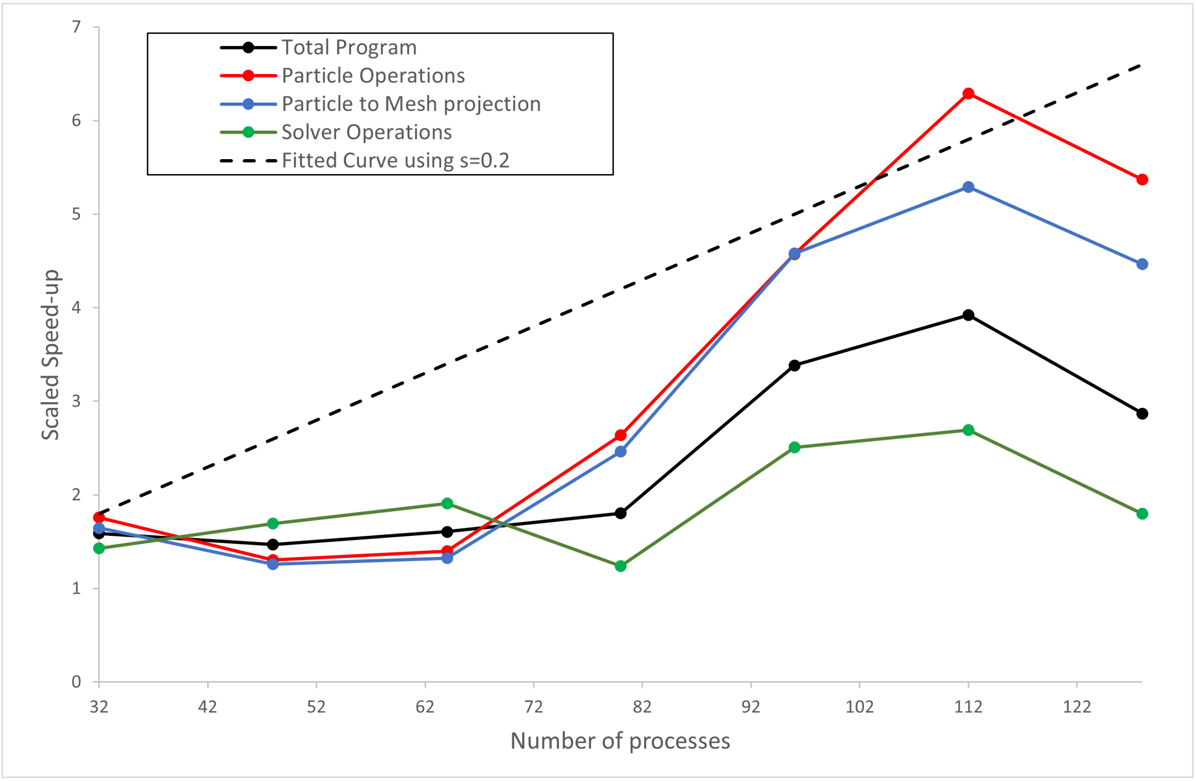


Fig. 5.13 Weak Scalability scaled speed-up of the different SL-PFEM parts.

5.2.3 Conclusions

The developed program shows a good strong scaling. In fact, by performing the strong scaling analysis, it permits to obtain a good estimate of the ratio nodes per process which allows to run the program in the most efficient manner. Above this ratio then resources are not being efficiently used and below means that more resources could be used.

Obviously the particles have to be taken into account, as the number of particles per element can increase the amount of resources needed to compute them. But as it has been shown that they have an even better strong scaling performance compared with the mesh, then a good starting point is by using the nodes as a reference.

The program is also analysed to check it's weak scaling in particular to see if the particle implementation is performing as expected. The results obtained are very promising and an analysis with large amounts of processes should be performed to truly see the program's weak scaling.

Chapter 6

Conclusions and future work

The objective of the thesis was the development of an HPC capable fluid dynamics solver that could solve a variety of problems with different types of interfaces such as fluid-fluid or fluid-solids. The method is based on a PFEM approach of the navier stokes equations initially proposed by [Ide+12; Ide+13] and further developed at the thesis [Bec14] where the basis of the PFEM are presented.

In a similar fashion to PFEM, the SL-PFEM proposed here uses a set of Lagrangian particles in order to solve the convective part of the Navier-Stokes equations. To solve the rest of the equations a fixed background FEM mesh is used. A set of operations are defined to map the information carried by the particles to the mesh and vice-versa. A projection operation is defined as the mapping of the particles to the mesh and an interpolation operation as the operation of passing back the information on the mesh to the particles.

Finally, an enriched method is applied when solving interface problems. This method, while not being as accurate as a refinement method, it allows to add new degrees of freedom at the interface without having to re-assemble the system of equations. In fact, a new SL-PFEM formulation of the level-set method is applied in order to accurately capture the interface.

6.1 Developments and summary of the applied strategies

The first part was to develop the algorithm to solve single phase flows, obtaining a second order accurate formulation of the SL-PFEM. This is developed in chapter 2, where the Navier-Stokes equations were separated in order to implement the SL-PFEM algorithm. The second order accurate scheme was obtained by applying the velocity Verlet integrator, a method that is second order accurate in time. Then a series of projection operators were presented, having two of them comply with the coherence condition. Finally, the FEM discretisation of the pressure and viscous terms of the Navier-Stokes equations was shown where an iterative variant of the fractional step method was implemented.
The last step was the analysis of the method to check that it is truly second order accurate in time and space. Therefore, an error analysis was performed on the whole scheme allowing to prove that the conjunction of second order spatial projectors and interpolators with the Velocity Verlet algorithm and the second order accurate FEM discretisation give a true second order formulation.

The second part was to implement an interface capturing model to calculate fluid-fluid interfaces. This was tackled in chapter 3 by first showing a new SL-PFEM formulation of the level-set method to obtain the interface. Once the interface is known, an enriched formulation of the Navier-Stokes equations was presented allowing to insert new degrees of freedom at the interface. This method, while being very similar to the refinement method, differs by locally collapsing the degrees of freedom which in turn allows to prevent the resizing of the system of equations. Chapter 4 continues exploring the interface problems, extending the formulation of the previous chapter allowing to solve fluid-solid interfaces and the combination of fluid-solid and fluid-fluid interfaces. A generic form of the enrichment method for the 2D and 3D problems was proposed and applied to the SL-PFEM method. This way, the enrichment method allows to have in a single element two interfaces at the same time.

The last part focuses on giving an introduction to how the scheme was implemented. It presents a brief summary of the methods used to optimise the program. The implementation of a domain decomposition library gives place to a highly parallel environment minimising the communication between processes. It also presents how the program scales in an HPC environment by using the two most typical variables: Strong Scaling and Weak Scaling. The implementation shows very good strong scaling but a weak scaling analysis proves to be difficult as large amounts of compute resources are needed.

But as an initial analysis, the particle side of the implementation shows promising weak scaling results.

6.2 Future work

In this thesis a tool that can solve fluid dynamics problems with different types of interfaces has been developed. It has laid the fundamental parts that can get a general problem running. But it has also left a wide variety of paths that can be taken to add new methodologies or even improve the existing scheme.

From an improvement stand point, the most obvious one would be implementation-wise. The current solver that has been implemented is not optimised for problems with domain-decomposition. General solvers such as "HYPRE" or "PETSC" have a general form of constructing and dividing the system of equations. This is done by dividing the rows of the system matrices through the processing units. This is not optimal as large amounts of data transfers must be made in order to change a system row, assemble right hand sides or even recover results. Solvers such as "FEMPAR" which are solvers based on domain decomposition could prove to be a really good fit with the current scheme.
Other improvements that can be done to the implementation would be to change the particle library. The program is designed in such a manner that to have various sets of particles proves to be difficult. Even worse, adding new data variables to the particle set also is a bit tedious. Therefore a new type of data-structure should be implemented that allows to add or remove as many variables to the particles.

With regard to new methodologies that can be implemented, one of them would be to implement the ability to solve seakeeping problems. The tool can solve moving objects within a fluid-fluid interface, but does not have an efficient way of implementing the wave-problem. Waves must be generated and must reach the object. This can take a long time to compute until the floating body starts reacting to the incoming waves. Therefore a way of initialising the problem with the waves rapidly would be extremely useful. In fact, this line of research is currently being developed by the naval hydrodynamics group of CIMNE.

Another line of research follows the previous line which would be to implement a turbulence model to the algorithm. The scheme that has been implemented serves well for most cases where turbulent flow is not predominant. But cases with high Reynolds number, an extremely fine mesh must be created if the current tool is used. The naval hydrodynamics world is right in the middle of this problem. Most of the cases that are analysed, such as an advancing ship, have large Reynolds numbers and to accurately capture the ship resistance a turbulence model is needed which allows to have larger mesh sizes and more real-world compute times.

By combining these last two implementations the current tool would truly be a seakeeping hydrodynamics tool. In fact, it would be a tool that can analyse various independent floating bodies at the same time while moving through different sea conditions.

References

[AIO05] R. Aubry, S. R. Idelsohn, and E. Oñate. “Particle finite element method in fluid-mechanics including thermal convection-diffusion”. In: *Computers and Structures*. 2005 (cit. on p. 4).

[Amd67] G. M. Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities”. In: *AFIPS Conference Proceedings - 1967 Spring Joint Computer Conference, AFIPS 1967*. Association for Computing Machinery, Inc, Apr. 1967, pp. 483–485 (cit. on pp. 170–172).

[Bak+11] A. H. Baker, R. D. Falgout, T. Gamblin, T. V. Kolev, M. Schulz, and U. M. Yang. “Scaling Algebraic Multigrid Solvers: On the Road to Exascale”. In: *Competence in High Performance Computing 2010* (2011), pp. 215–226 (cit. on pp. 173, 174).

[Bec14] P. A. Becker. “An enhanced Particle Finite Element Method with special emphasis on debris flows”. In: (2014) (cit. on pp. 5, 46, 179).

[Bih+16] H. Bihs, A. Kamath, M. Alagan Chella, A. Aggarwal, and Ø. A. Arntsen. “A new level set numerical wave tank with improved density interpolation for complex wave hydrodynamics”. In: *Computers and Fluids* 140 (2016), pp. 191–208 (cit. on p. 2).

[BIO15] P. Becker, S. R. Idelsohn, and E. Oñate. “A unified monolithic approach for multi-fluid flows and fluid–structure interaction using the Particle Finite Element Method with fixed mesh”. In: *Computational Mechanics* 55.6 (June 2015), pp. 1091–1104 (cit. on p. 5).

[BP98] O. Botella and R. Peyret. “Benchmark spectral results on the lid-driven cavity flow”. In: *Computers & Fluids* 27.4 (May 1998), pp. 421–433 (cit. on pp. 49, 51).

[Car+07] P. M. Carrica, R. V. Wilson, R. W. Noack, and F. Stern. “Ship motions using single-phase level set with dynamic overset grids”. In: *Computers & Fluids* 36.9 (Nov. 2007), pp. 1415–1433 (cit. on p. 1).

[Cas+11] T. Castiglione, F. Stern, S. Bova, and M. Kandasamy. “Numerical investigation of the seakeeping behavior of a catamaran advancing in regular head waves”. In: *Ocean Engineering* 38.16 (Nov. 2011), pp. 1806–1822 (cit. on p. 1).

[CB00] R. Codina and J. Blasco. “Stabilized finite element method for the transient Navier–Stokes equations based on a pressure gradient projection”. In: *Computer Methods in Applied Mechanics and Engineering* 182.3-4 (Feb. 2000), pp. 277–300 (cit. on p. 33).

[CB97] R. Codina and J. Blasco. “A finite element formulation for the Stokes problem allowing equal velocity-pressure interpolation”. In: *Computer Methods* *in Applied Mechanics and Engineering* 143.3-4 (1997), pp. 373–391 (cit. on p. 33).

[Cha+96] Y. C. Chang, T. Y. Hou, B. Merriman, and S. Osher. “A level set formulation of Eulerian interface capturing methods for incompressible fluid flows”. In: *Journal of Computational Physics* 124.2 (Mar. 1996), pp. 449–464 (cit. on p. 2).

[Cola] J. Colom-Cobb. *2D Sloshin in Tank - Experimental comparison for frequency 4.34 rad/s* (cit. on p. 117).

[Colb] J. Colom-Cobb. *2D Sloshin in Tank - Experimental comparison for frequency 4.87 rad/s* (cit. on p. 117).

[Col+20] J. Colom-Cobb, J. Garcia-Espinosa, B. Servan-Camas, and P. Nadukandi. “A second-order semi-Lagrangian particle finite element method for fluid flows”. In: *Computational Particle Mechanics* 7.1 (2020), pp. 3–18 (cit. on p. 7).

[EGH00] R. Eymard, T. Galllouët, and R. Herbin. *Finite volume methods*. Jan. 2000 (cit. on p. 3).

[FUI05] T. Fujiwara, M. Ueno, and Y. Ikeda. “A New Estimation Method of Wind Forces and Moments acting on Ships on the basis of Physical Component Models”. In: *Journal of the Japan Society of Naval Architects and Ocean Engineers* 2.0 (2005), pp. 243–255 (cit. on pp. 75, 78).

[Fur] P. Furijaz. *Patrol Torpedo Boat aka PT-Boat | 3D CAD Model Library | GrabCAD* (cit. on p. 152).

[Gar+20] J. García-Espinosa, E. Oñate, B. Serván-Camas, M. A. Celigueta, S. Latorre, and J. Colom-Cobb. “Development of New Lagrangian Computational Methods for Ice-Ship Interaction Problems: NICESHIP Project”. In: *Computational Methods in Applied Sciences*. Vol. 54. Springer, 2020, pp. 121–153 (cit. on p. 7).

[GG15] J. M. Gimenez and L. M. González. “An extended validation of the last generation of particle finite element method for free surface flows”. In: *Journal of Computational Physics* (2015) (cit. on p. 5).

[GGF19] L. M. González-Gutierrez, J. M. Gimenez, and E. Ferrer. “Instability onset for submerged cylinders”. In: *Physics of Fluids* 31.1 (Jan. 2019) (cit. on pp. 5, 123, 125, 126).

[Gim+16] J. M. Gimenez, P. Morin, N. Nigro, and S. Idelsohn. “Numerical Comparison of the Particle Finite Element Method Against an Eulerian Formulation”. In: *Advances in Computational Fluid-Structure Interaction and Flow Simulation*. Springer International Publishing, 2016, pp. 7–24 (cit. on p. 5).

[Gim+17] J. M. Gimenez, D. E. Ramajo, S. Márquez Damián, N. M. Nigro, and S. R. Idelsohn. “An assessment of the potential of PFEM-2 for solving long real-time industrial applications”. In: *Computational Particle Mechanics* 4.3 (July 2017), pp. 251–267 (cit. on p. 5).

[Gim+19] J. M. Gimenez, H. J. Aguerre, S. R. Idelsohn, and N. M. Nigro. “A second-order in time and space particle-based method to solve flow problems on arbitrary meshes”. In: *Journal of Computational Physics* 380 (Mar. 2019), pp. 295–310 (cit. on p. 5).

[GM82] R. A. Gingold and J. J. Monaghan. “Kernel estimates as a basis for general particle methods in hydrodynamics”. In: *Journal of Computational Physics* (1982) (cit. on p. 3).

[Gon01] L. M. González Gutierrez. “Integración de las ecuaciones de Navier-Stokes mediante el método de los elementos finitos y el método de las características : Aplicaciones a casos con superficie libre”. PhD thesis. 2001 (cit. on p. 5).

[Gus88] J. L. Gustafson. “Reevaluating amdahl’s law”. In: *Communications of the ACM* 31.5 (May 1988), pp. 532–533 (cit. on pp. 173, 176).

[Her09] C. O. Herbert. “A Finite Element Model for Free Surface and Two Fluid Flows on Fixed Meshes”. PhD thesis. July 2009 (cit. on pp. 2, 81, 82, 85).

[HLW03] E. Hairer, C. Lubich, and G. Wanner. “Geometric numerical integration illustrated by the Störmer–Verlet method”. In: *Acta Numerica* 12 (May 2003), pp. 399–450 (cit. on p. 19).

[HN81] C. W. Hirt and B. D. Nichols. “Volume of fluid (VOF) method for the dynamics of free boundaries”. In: *Journal of Computational Physics* 39.1 (Jan. 1981), pp. 201–225 (cit. on p. 2).

[HS67] J. L. Hess and A. M. Smith. “Calculation of potential flow about arbitrary bodies”. In: *Progress in Aerospace Sciences* 8.C (Jan. 1967), pp. 1–138 (cit. on p. 1).

[ICO03] S. R. Idelsohn, N. Calvo, and E. Onate. “Polyhedrization of an arbitrary 3D point set”. In: *Computer Methods in Applied Mechanics and Engineering* 192.22-23 (June 2003), pp. 2649–2667 (cit. on p. 4).

[Ide+] S. R. Idelsohn, E. Oñate+, N. Calvo, and F. D. Pin. *The meshless finite element method*. Tech. rep. (cit. on p. 4).

[Ide+06a] S. Idelsohn, E. Oñate, F. D. Pin, and N. Calvo. “Fluid–structure interaction using the particle finite element method”. In: *Computer Methods in Applied Mechanics and Engineering* 195.17-18 (Mar. 2006), pp. 2100–2123 (cit. on pp. 3, 4, 129).

[Ide+06b] S. R. Idelsohn, E. Oñate, F. Del Pin, and N. Calvo. “Fluid-structure interaction using the particle finite element method”. In: *Computer Methods in Applied Mechanics and Engineering* 195.17-18 (Mar. 2006), pp. 2100–2123 (cit. on p. 3).

[Ide+08] S. R. Idelsohn, J. Marti, A. Limache, and E. Oñate. “Unified Lagrangian formulation for elastic solids and incompressible fluids: Application to fluid-structure interaction problems via the PFEM”. In: *Computer Methods in Applied Mechanics and Engineering* 197.19-20 (Mar. 2008), pp. 1762–1776 (cit. on pp. 3, 4).

[Ide+12] S. Idelsohn, N. Nigro, A. Limache, and E. Oñate. “Large time-step explicit integration method for solving problems with dominant convection”. In: *Computer Methods in Applied Mechanics and Engineering* 217-220 (Apr. 2012), pp. 168–185 (cit. on pp. 5, 179).

[Ide+13] S. R. Idelsohn, N. M. Nigro, J. M. Gimenez, R. Rossi, and J. M. Marti. “A fast and accurate method to solve the incompressible Navier-Stokes equations”. In: *Engineering Computations (Swansea, Wales)*. Vol. 30. 2. 2013, pp. 197–222 (cit. on pp. 5, 179).

[Ide+14] S. R. Idelsohn, J. Marti, P. Becker, and E. Oñate. “Analysis of multifluid flows with large time steps using the particle finite element method”. In: *International Journal for Numerical Methods in Fluids* 75.9 (July 2014), pp. 621–644 (cit. on p. 5).

[Ide+15] S. Idelsohn, E. Oñate, N. Nigro, P. Becker, and J. Gimenez. "Lagrangian versus Eulerian integration errors". In: *Computer Methods in Applied Mechanics and Engineering* 293 (2015), pp. 191–206 (cit. on p. 5).

[IOD03] S. R. Idelsohn, E. Oñate, and F. Del Pin. "A Lagrangian meshless finite element method applied to fluid-structure interaction problems". In: *Computers and Structures* 81.8-11 (May 2003), pp. 655–671 (cit. on p. 4).

[IOP04] S. Idelsohn, E. Oñate, and F. D. Pin. "The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves". In: *International Journal for Numerical Methods in Engineering* 61.7 (Oct. 2004), pp. 964–989 (cit. on pp. 3, 4).

[Itō77] S. Itō. "Study of the transient heave oscillation of a floating cylinder." PhD thesis. Massachusetts Institute of Technology, 1977 (cit. on pp. 149, 150, 152).

[ITT14] ITTC. "ITTC – Recommended Procedures and Guidelines - Analysis of speed/power trial data. 7.5-04-01-01.2 (Revision 01)". In: (2014), p. 33 (cit. on p. 75).

[ITT75] ITTC. *14th International Towing Tank Conference 1975 Report of the resistance commitee.* Tech. rep. 1975 (cit. on p. 121).

[ITT85] ITTC. *Collected experimental resistance component and flow data for three surface ship model hulls, 17TH IITC conference.* Tech. rep. 202. DAVID W. TAYLOR NAVAL SHIP RESEARCH and DEVELOPMENT CENTER, 1985 (cit. on p. 119).

[JC17] H. Jiang and L. Cheng. "Strouhal–Reynolds number relationship for flow past a circular cylinder". In: *Journal of Fluid Mechanics* 832 (Dec. 2017), pp. 170–188 (cit. on pp. 63, 65–67, 69, 71, 73, 149).

[Kar20] G. Karypis. *METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering / Karypis Lab.* 2020 (cit. on p. 165).

[KTO95] S. Koshizuka, H. Tamako, and Y. Oka. *a Particle Method for Incompressible Viscous Flow With Fluid Fragmentation.* 1995 (cit. on p. 4).

[Lan+11] F. M. Lang, A. S. Iglesias, M. Antuono, and A. Colagrossi. "Benefits of using a Wendland Kernel for free-surface flows". In: *Proceedings of 6th ERCOFTAC SPHERIC workshop on SPH applications.* 2011, pp. 30–37 (cit. on p. 22).

[Lar+08] A. Larese, R. Rossi, E. Oñate, and S. R. Idelsohn. "Validation of the particle finite element method (PFEM) for simulation of free surface flows". In: *Engineering Computations (Swansea, Wales)* (2008) (cit. on p. 4).

[LL03] G. R. Liu and M. B. Liu. *Smoothed Particle Hydrodynamics.* WORLD SCIENTIFIC, Oct. 2003 (cit. on p. 4).

[LL10] M. B. Liu and G. R. Liu. "Smoothed particle hydrodynamics (SPH): An overview and recent developments". In: *Archives of Computational Methods in Engineering* 17.1 (Mar. 2010), pp. 25–76 (cit. on pp. 3, 4).

[LLN21] LLNL. *HYPRE - Lawrence Livermore National Laboratory.* 2021 (cit. on p. 173).

[MCS10] S. M. Mousaviraad, P. M. Carrica, and F. Stern. "Development and validation of harmonic wave group single-run procedure for RAO with comparison to regular wave and transient wave group procedures using URANS". In: *Ocean Engineering* 37.8-9 (June 2010), pp. 653–666 (cit. on p. 1).

[Mon94] J. J. Monaghan. “Simulating free surface flows with SPH”. In: *Journal of Computational Physics* 110.2 (Feb. 1994), pp. 399–406 (cit. on pp. 3, 4).

[Nad+17] P. Nadukandi, B. Servan-Camas, P. A. Becker, and J. Garcia-Espinosa. “Seakeeping with the semi-Lagrangian particle finite element method”. In: *Computational Particle Mechanics* 4.3 (July 2017), pp. 321–329 (cit. on pp. 5, 6, 43).

[New85] J. N. Newman. “Algorithms for the free-surface Green function”. In: *Journal of Engineering Mathematics* 19.1 (Mar. 1985), pp. 57–67 (cit. on p. 1).

[New92] J. N. Newman. “The approximation of free-surface Green functions”. In: *Wave Asymptotics*. Ed. by P. A. Martin and G. R. Wickham. Cambridge University Press, 1992, pp. 107–135 (cit. on p. 1).

[OF01] S. Osher and R. P. Fedkiw. “Level Set Methods: An Overview and Some Recent Results”. In: *Journal of Computational Physics* 169.2 (May 2001), pp. 463–502 (cit. on p. 2).

[OFC14] E. Oñate, A. Franci, and J. M. Carbonell. “A particle finite element method for analysis of industrial forming processes”. In: *Computational Mechanics* 54.1 (July 2014), pp. 85–107 (cit. on p. 4).

[OM00] E. Oñate and M. Manzan. *Stabilization techniques for finite element analysis of convection-diffusion problems*. CIMNE, 2000 (cit. on p. 10).

[Oña+06] E. Oñate, J. García, S. Idelsohn, and F. D. Pin. “Finite calculus formulations for finite element analysis of incompressible flows. Eulerian, ALE and Lagrangian approaches”. In: *Computer Methods in Applied Mechanics and Engineering* 195.23-24 (Apr. 2006), pp. 3001–3037 (cit. on p. 129).

[Oña+11] E. Oñate, S. R. Idelsohn, M. A. Celigueta, R. Rossi, J. Marti, J. M. Carbonell, P. Ryzhakov, and B. Suárez. “Advances in the Particle Finite Element Method (PFEM) for Solving Coupled Problems in Engineering”. In: *Particle-Based Methods*. Springer Netherlands, 2011, pp. 1–49 (cit. on pp. 3, 129).

[Oña+96a] E. Oñate, S. Idelsohn, O. C. Zienkiewicz, and R. L. Taylor. “A finite point method in computational mechanics. Applications to convective transport and fluid flow”. In: *International Journal for Numerical Methods in Engineering* (1996) (cit. on p. 4).

[Oña+96b] E. Oñate, S. Idelsohn, O. C. Zienkiewicz, R. L. Taylor, and C. Sacco. “A stabilized finite point method for analysis of fluid mechanics problems”. In: *Computer Methods in Applied Mechanics and Engineering* (1996) (cit. on p. 4).

[Ope] Openfoam. *OpenFOAM® - Official home of The Open Source Computational Fluid Dynamics (CFD) Toolbox* (cit. on p. 3).

[OS88] S. Osher and J. A. Sethian. “Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations”. In: *Journal of Computational Physics* 79.1 (Nov. 1988), pp. 12–49 (cit. on p. 2).

[Pin+07] F. D. Pin, S. Idelsohn, E. Oñate, and R. Aubry. “The ALE/Lagrangian Particle Finite Element Method: A new approach to computation of free-surface flows and fluid-object interactions”. In: *Computers and Fluids* (2007) (cit. on p. 4).

[RD53] A. Roshko and J. H. De Leeuw. *On the development of turbulent wakes from vortex streets*. Tech. rep. Wahsington, D.C.: National Advisory Committee for Aeronautics., 1953 (cit. on pp. 63, 65, 149).

[Ser16] B. Serván-Camas. “A time-domain finite element method for seakeeping and wave resistance problems”. In: 10.20868/UPM.thesis.39794 (2016) (cit. on pp. 1, 107, 119).

[Set01] J. A. Sethian. “Evolution, Implementation, and Application of Level Set and Fast Marching Methods for Advancing Fronts”. In: *Journal of Computational Physics* 169.2 (May 2001), pp. 503–555 (cit. on p. 2).

[SG13a] B. Serván-Camas and J. García-Espinosa. “Accelerated 3D multi-body seakeeping simulations using unstructured finite elements”. In: *Journal of Computational Physics* 252 (Nov. 2013), pp. 382–403 (cit. on p. 1).

[SG13b] B. Serván-Camas and J. García-Espinosa. “Accelerated 3D multi-body seakeeping simulations using unstructured finite elements”. In: *Journal of Computational Physics* 252 (Nov. 2013), pp. 382–403 (cit. on p. 107).

[SG17] H. Schlichting and K. Gersten. *Boundary-Layer Theory*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017 (cit. on p. 65).

[Sim+13] C. D. Simonsen, J. F. Otzen, S. Joncquez, and F. Stern. “EFD and CFD for KCS heaving and pitching in regular head waves”. In: *Journal of Marine Science and Technology* 18.4 (Dec. 2013), pp. 435–459 (cit. on p. 1).

[SLC01] O. Soto, R. Lohner, and J. Cebral. “An implicit monolithic time accurate finite element scheme for incompressible flow problems”. In: *15th AIAA Computational Fluid Dynamics Conference* June (2001) (cit. on pp. 27, 33).

[Smi] R. Smith. *Open Dynamics Engine* (cit. on p. 135).

[Sou+06] A. Souto-Iglesias, L. Delorme, L. Pérez-Rojas, and S. Abril-Pérez. “Liquid moment amplitude assessment in sloshing type problems with smooth particle hydrodynamics”. In: *Ocean Engineering* 33.11-12 (2006), pp. 1462–1484 (cit. on pp. 114, 115).

[SS03] J. A. Sethian and P. Smereka. “Level Set Methods for Fuid Interfaces”. In: *Annual Review of Fluid Mechanics* 35.1 (Jan. 2003), pp. 341–372 (cit. on p. 2).

[SS68] SHEPARD D and D. Shepard. “A two-dimensional interpolation function for irregularly-spaced data”. In: *Proceedings of the 1968 23rd ACM national conference on -*. New York, New York, USA: ACM Press, 1968, pp. 517–524 (cit. on p. 22).

[Sus94] M. Sussman. “A level set approach for computing solutions to incompressible two-phase flow”. In: *Journal of Computational Physics* 114.1 (Sept. 1994), pp. 146–159 (cit. on p. 3).

[Swo+82] W. C. Swope, H. C. Andersen, P. H. Berens, and K. R. Wilson. “A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters”. In: *The Journal of Chemical Physics* 76.1 (Jan. 1982), pp. 637–649 (cit. on p. 19).

[Ver67] L. Verlet. “Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules”. In: *Physical Review* 159.1 (July 1967), pp. 98–103 (cit. on p. 19).

[WAM] WAMIT. *Wamit User Manual* (cit. on p. 1).

[Wel+98] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in Physics* 12.6 (Dec. 1998), p. 620 (cit. on p. 3).

[Wen95] H. Wendland. “Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree”. In: *Advances in Computational Mathematics* 4.1 (Dec. 1995), pp. 389–396 (cit. on p. 22).

[ZT02] O. C. Zienkiewicz and R. L. Taylor. “The finite element method. Fifth edition (O. C. Zienkiewicz, R. L. Taylor)”. In: *Bautechnik* 79.2 (Feb. 2002), pp. 122–123 (cit. on p. 2).

[ZTN13] O. C. Zienkiewicz, R. L. Taylor, and P. Nithiarasu. *The Finite Element Method for Fluid Dynamics: Seventh Edition*. Elsevier Ltd, 2013, pp. 1–544 (cit. on p. 2).