A Finite Element Model for Free Surface and Two Fluid Flows on Fixed Meshes

Herbert Coppola Owen

Advisor: **Ramon Codina**

Escola Tècnica Superior d' Enginyers de Camins, Canals i Ports

Universitat Politècnica de Catalunya

April 2009

A mi familia,

ACTA DE QUALIFICACIÓ DE LA TESI DOCTORAL

Reunit el tribunal integrat pels sota signants per jutjar la tesi doctoral:

Títol de la tesi: A Finite Element Model for Free Surface and Two Fluid Flows on Fixed Meshes

Autor de la tesi: Angel Heriberto Coppola Owen

Acorda atorgar la qualificació de:

☐ No apte

☐ Aprovat

☐ Notable

☐ Excel·lent

☐ Excel·lent Cum Laude

Barcelona, ... de/d' ..... de .....

El President	El Secretari
.....	.....
(nom i cognoms)	(nom i cognoms)

El vocal	El vocal	El vocal
.....	.....	.....
(nom i cognoms)	(nom i cognoms)	(nom i cognoms)

Acknowledgments

Ramon, no doubt I have been lucky. Not only is he brilliant but also kind, fair and patient. Thank you Ramon!

I would also like to thank the RMEE department of the Universitat Politècnica de Catalunuya and CIMNE. Specially, Eugenio Oñate, for giving me the opportunity of coming to Barcelona. Working with Guillaume Houzeaux, Santi Badia, Oriol Guasch, Matias Avila, Christian Muñoz Joan Baiges, Shu-Ren Hysing, Noel Hernandez, and Javier Principe has been very enriching. Special thanks go to Javier with whom I worked both at Buenos Aires and Barcelona. He has helped me with both theoretical and practical matters. With Ramon, Guillaume, Javier and Mariano Vazquez we started the code I have used in this thesis. It has been very nice to do some team work. I have also learned from my interaction with Gerardo Valdez, Romain Aubry, Monica de Mier, Ricardo Rossi, Carlos Labra, Pooyan Dadvan and Vicente. Thanks to the rest of my office mates: Roberto, Jeovan, Pablo and Maritzabel.

During my thesis I have worked with the people from Quantech in mould filling simulations. Special thanks go to Martin Solina with whom I have interacted most.

Rainald Lohner received me for a stay at George Mason University. I would like to thank him and his group: Fernando Camelli, Juan Cebral, Chi Yang, Fernando Mut, Marcelo Castro, Joaquin Arteaga, Romain Aubry and Orlando Soto. Cielo, Martin and the rest of the Colombian-Slovak group made our stay at USA more pleasant.

Marcela Goldschmit and Eduardo Dvorkin introduced me to computational mechanics, allowed me to work with them at the Center for Industrial Research (CINI) and motivated me to pursue doctoral studies abroad. I would like to thank them and all

of my companions at CINI.

The financial support received from the Agència de Gestió d'Ajuts Universitaris i de Recerca of the *Generalitat de Catalunya* (Catalan Government) and the *European Social Fund* through a doctoral grant is acknowledged.

Finally, I would like to thank my friends in Barcelona, Stevie, Riso, Vale, Bea, Daniel, Adrian, Sergio and Joseph and the UPC rowing team who have helped me disconnect from the doctorate. I also would like to mention my friends from St. John's and the University of Buenos Aires that, despite the distance, are always present. The support of my family and the love of wife, Liliana, have helped me during the good and bad times.

Abstract

Flows with moving interfaces (free surface and two-fluid interface problems) appear in numerous engineering applications. The methods presented in this thesis are oriented mainly to the simulation of mould filling process. Nevertheless the methodology is sufficiently general as to be applied to most free surface and two-fluid interface flows. Numerical modeling provides an efficient way of analyzing the physical phenomena that occur during casting and injection processes. It gives insight into details of the flow that would otherwise be difficult to observe.

A fixed mesh finite element method, where the interface position is captured by the Level Set function, is used. Low Froude number flows are particularly challenging for fixed grid methods. An accurate representation is needed in the elements cut by the interface for such flows. Two alternatives are proposed.

The first alternative is to use the typical two-phase flow model enriching the pressure shape functions so that the discontinuity in the pressure gradient at the interface can be better approximated. The improvement in the representation of the pressure gradient is shown to be the key to ingredient for the successful modeling of such flows.

The influence of the second fluid can be neglected on a wide range of applications to end up with a free surface model that is simpler than the two-phase flow model. The discontinuity in the pressure gradient disappears because only one fluid is simulated. The particularity of this second approach is that a fixed mesh is used. Boundary conditions are applied accurately using enhanced integration and integrating only in the filled part of cut elements. A fixed mesh ALE approach is developed to correctly take into account that the domain is moving despite a fixed mesh is used.

Pressure segregation methods are explored as an alternative to the monolithic discretization of the Navier Stokes equations. They uncouple the velocity and pressure unknowns, leading to smaller and better conditioned subproblems. Pressure correction and velocity correction methods are presented and compared numerically. Using a discrete Laplacian a numerically stable third order velocity correction method is obtained.

The methods are applied to three dimensional mould filling problems borrowed directly from the foundry with very satisfactory results. The free surface monolithic model turns out to be the most robust and efficient option. The comparison with a commercial code shows the accuracy and efficiency of the method we propose.

Contents

Chapter 1

Introduction and basic model

1.1 Introduction

Flows with moving interfaces (free surface and two–fluid interface problems) appear in numerous engineering applications. The numerical simulation of interface flows can be a great ally in the understanding and improvement of such applications. The great number of publications on the subject is the best evidence of the interest on the subject. The fields of application are as wide as can be observed from the following examples: drop formation in ink-jet devices [104], ship hydrodynamics [76–78, 88] and mould filling [31, 80, 97].

The methods presented in this thesis will be oriented mainly to the simulation of mould filling processes. Nevertheless the methodology is sufficiently general as to be applied to most free surface and two–fluid interface flows. Numerical modeling provides an efficient way of analyzing the physical phenomena that occur during casting and injection processes. It gives insight into details of the flow that would otherwise be difficult to observe. When coupled with the appropriate models, it can also provide information about heat transfer and solidification. The numerical results can help shorten the design process and optimize casting parameters to improve the castings, reduce scrap and use less energy.

In this thesis we will deal with both free surface and two-fluid interface problems. The

former are a special case of the latter where the influence of one of the fluids on the other one is negligible. In most casting applications the free surface model can be used because one is only interested in the behavior of the fluid and the influence of the air is negligible. Obviously free surface flows can be modeled as two fluid flows where the properties of one of the fluids are much smaller than those of the other one. Special models that take into account the particularities of free surface flows can also be developed (see Chapter 3). The term interface flows refers to both free surface and two–fluid interface flows.

The objective of this thesis is to develop or improve techniques that can be used in finite element mould filling software. The range of numerical methods available for interface flows is as wide as the range of applications. As in most CFD applications, several spatial discretization methods can be used, among them: finite differences, finite elements, finite volumes and even meshless methods. On the other hand the existence of a moving interface gives raise to a huge number of methods to deal with such flows. In the next subsection we will present a brief classification of the most relevant ones. The first and perhaps the most significant classification depends on the nature, fixed or moving, of the grid used. In this thesis we will use a fixed mesh approach, in particular the Level Set method. Both the discretization method and the fixed grid approach were selections made prior to the beginning of this thesis. The choice of the best discretization method is a problem dependent question that we do not intend to answer in this thesis. The classification of the different moving interface methods presented in the next subsection intends to clarify where we stand and show some of the alternatives we could have, something we hope will be useful for the reader that steps into the subject. On the other hand, we hope that it can show that the methodology we will work with is a pretty reasonable choice.

1.1.1 Classification of methods for flows with interfaces

The classification of the methods used for free surface and two fluid flows is not an easy task mainly because of the wide range of schemes that exist. Some interesting

classifications and comparisons can be found in [70, 103, 109, 112, 116]

As we have already mentioned one of the classifications depends on the nature, *fixed* or *moving*, of the grid used. Another common option is to classify methods into interface *tracking* and interface *capturing* [70]. In *tracking* schemes the position of discrete points x_i lying on the interface is tracked for all time by integrating the evolution equation

$$\frac{d\boldsymbol{x}_i}{dt} = \boldsymbol{u}_i$$

where $\boldsymbol{u}_i$ is the velocity with which interface point $\boldsymbol{x}_i$ moves Moving mesh methods are interface tracking schemes where the points i correspond to nodes placed on the interface. In *capturing* methods, the interface is not explicitly tracked, but rather captured using some interface function (ψ) defined over the whole mesh that allows to determine which fluid occupies any point in the domain. The evolution equation for the interface function is given by

$$\frac{\partial \psi}{\partial t} + (\boldsymbol{u} \cdot \nabla)\, \psi = 0.$$

The third classification would separate methods into *Eulerian* ones which solve the Navier–Stokes equations on fixed grids and *Lagrangian* (or Arbitrary Eulerian Lagrangian, ALE) ones which solve them on a grid that follows (or partially follows) the characteristics of the flow.

In order to try to unify the three previous classifications one could speak about *moving mesh, interface tracking* or *Lagrangian* schemes and *fixed mesh, interface capturing* or *Eulerian* ones. Despite this might seem the most natural way of unifying the previous classifications, there are some methods that would not fit properly into such unification and could be considered as an exception to the rule. In the pursuit for better methods it is not uncommon to see authors that try to blend components from the two main class of methods we have defined. For example Front Tracking Methods [112] which have their roots in the MAC method of Harlow and Welch [54] are, as their name indicates, tracking schemes but they use a fixed mesh to model the flow. In Chapter 3 we will present a model for free surface flow that uses a ALE approach on a fixed mesh and thus, would on one hand be classified into the Lagrangian group and on the other into the fixed mesh

group.

Neglecting some particular schemes, the unified classification we have presented can be considered valid for most cases. In most interface capturing techniques a fixed computational domain is used and an interface function is used to capture the position of the interface. The interface is captured within the resolution of the fixed mesh and the boundary conditions at the interface are somehow approximated. In most interface tracking techniques the mesh is updated in order to track the interface. The simplest approach is to deform the mesh without changing its topology, but it is valid only for very simple flows. As the flow becomes more complex and unsteady remeshing and consequently the projection of the results from the old to the new mesh are needed [3, 64, 71, 87].

For the same mesh size moving grid techniques lead to a more accurate representation of the interface at a higher computational cost. In Chapter 2 we will use a fixed grid method and introduce modifications to the basic formulation to enhance the representation of the flow at the interface. The idea of enriching the representation of an unknown at a material discontinuity is not new and several approaches can be found in the literature [19, 81].

Fixed mesh methods generally share two basic steps, one where the motion in both phases is found as the solution of the Navier–Stokes equations with variable properties and the other one, where an equation for an interface function that allows to determine the position of the interface, and thus the properties to be assigned in the previous step, is solved. The different methods differ mainly in the method used to determine the position of the interface but also differences can be found in the way to approximate the properties to be used close to the interface. In Section 2 we will deal with the first step and in Section 3 with the second one.

A mentioned previously, we capture the interface using the so called *level set* method (see [18, 106] and [89, 90, 104] for an overview), also called *pseudo–concentration technique* [110] and very similar to the volume of fluid (VOF) technique [57,79]. This formulation has been widely used to track free surfaces in mould filling (see for example [31, 73, 80, 94, 97],

among other references) and other metal forming processes.

1.1.2 Organization

This thesis is organized as follows. The present Chapter presents an introduction to the numerical simulation of free surface and two–fluid interface flows and the basic model used to simulate interface flows on fixed meshes. Also in the next subsection some preliminary or notation issues will be included.

The next Section deals with the solution of the Navier–Stokes equations for flows with interfaces. First the equations to be solved are presented. Then their space and time discretization is described. Finally two stabilization techniques that allow us to model flows with important convective effects and also enable the use of equal order finite element interpolations for the velocity and pressure are introduced. A Monolithic or Mixed discretization is used.

The third Section deals with the Level Set Method used to determine the position of the front. The relations with some of the other most popular interface capturing techniques, pseudo-concentration and VOF, are analyzed. The space and time discretization of the Level Set equation is undertaken and the problem is stabilized. Finally some technical issues such a reinitialization and calculation of extension velocities are briefly discussed.

This first Chapters describes the basic elements of a typical Finite Element Level Set model for interface flows. The next two Chapters present developments we propose to improve the simulation of two phase flows. These developments gain special importance in the simulation of low Froude number flows, that is, when the gravitational forces are bigger than the inertial ones. Something we would like to remark is that our improvements are focused on the modelling of the Navier–Stokes equations and not on the step that deals with the Level Set equation. The poor behavior that can be observed using the typical model for low Froude number flows and the degree of improvement we have obtained justify such choice. Strangely, specially in the level set community, much more attention

is paid to the solution of the Level Set equation than to the solution of the two fluid Navier–Stokes equations.

Chapter 2 presents our first original contribution [36,37], a way of improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions. Chapter 3 presents our second important contribution [29,38], another way of improving interface flows that is applicable only to free surface flows. An ALE formulation is used but the mesh remains fixed (FM-ALE).

Taking into account that the objective of our research is to be used efficiently in finite element mould filling software, we try to concentrate on those items we find hinder our objective most. The most relevant one is the size of the problems we can handle and the efficiency with which we can tackle them. The step that solves the Navier-Stokes equations is by far more computationally expensive than the one that solves the Level Set equations and is therefore the one we wish to improve. When solving the Navier-Stokes equations the most expensive step, specially as the size of the problem grows, is the solution of the resulting linear system. Two types of solvers are available, direct and iterative. The former have the advantage that the obtention of a solution is guaranteed after a fixed number of steps that does not depend on the condition of the matrix of the system to be solved. The latter, despite their convergence is not guaranteed in practical situations, as it depends on the condition number of the matrix to be solved, have the advantage that the computational cost increases much more slowly than that of direct solvers as the size of the system to be solved increases. Therefore they are the undisputed option as the size of the systems to be solved grows, as happens in industrial 3D mould filling simulations. Our initial experience with iterative solvers was not very satisfactory and most of our problems were solved using direct solvers. Despite we implemented and tested a modern sparse direct solver called MUMPS [1,2] that brought about significant improvements with respect to our previous direct solver, it is still too expensive for real industrial problems. After making some further experience with iterative solvers and their preconditioners we have obtained much better results as shown in Chapter 5.

In Chapter 4 we explore pressure segregation methods [4] (also known as Fractional

Step methods) for the Navier–Stokes equations. Since their appearance in the late 1960's, with the pioneering works of Chorin [20] and Teman [107], these methods have enjoyed widespread popularity. Their common feature is the decoupling of the velocity and pressure interpolation. Such uncoupling yields an important computational cost reduction, on one hand because the systems are uncoupled, and on the other, perhaps the key one, because each of the resulting systems are better conditioned than the one resulting from the monolithic system. Both pressure correction and velocity correction methods (a more recent option) will be explored. Besides, the predictor corrector versions will also be tested. Predictor corrector methods also decouple the solution of the velocity and pressure, but they iterate until convergence so as to recover the monolithic solution. Fractional Step methods can be seen as a predictor corrector scheme that is only allowed to iterate once. The choice between monolithic or pressure segregation schemes is, up to what we understand, an open question and there are important research groups that stick to one or the other formulation. Obviously it is also a problem dependent question. Our intention is to build some solid knowledge on which to base our selection (for the problems we are interested in) resorting mainly to numerical experimentation.

In Chapter 5 we apply the tools developed in the previous Chapters to mould filling problems. The free surface model is compared against the enriched pressure two phase model. The results obtained with the monolithic scheme are compared against the ones obtained with the velocity correction scheme. Moreover the results are compared against the ones obtained with a commercial code. The advantages introduced by the two models we propose are clearly noticeable on low Froude number flows.

1.1.3 Notation Issues

Functional Spaces

In order to introduce the notation to be used in this work, a brief summary of some concepts on functional analysis will be presented. For a more detailed presentation any standard text on the subject can be consulted [85]

Let $\Omega \subset \mathbb{R}^d$, $d = 2$ or 3, be a bounded domain. $\mathcal{C}_0^\infty(\Omega)$ is the set of infinitely differentiable real functions with compact support on Ω. $L^p(\Omega), 1 \leq p < \infty$ is the space of real functions defined on Ω with p-th power absolutely integrable with respect to the Lebesgue measure. It is a Banach space with the associated norm

$$\|u\|_{L^p(\Omega)} := \left(\int_\Omega |u(x)|^p \, \mathrm{d}\Omega \right)^{1/p}.$$

L^2 is of special interest since it is a Hilbert space endowed with the scalar product

$$(u, v)_\Omega = \int_\Omega u(x) \, v(x) \, \mathrm{d}\Omega$$

and the norm

$$\|u\|_{L^2(\Omega)} := (u, u)_\Omega^{1/2}.$$

The Sobolev space $W^{m,p}(\Omega)$ is the space of functions in $L^p(\Omega)$ whose weak derivatives of order less than or equal m belong to $L^p(\Omega)$, being m an integer and $1 \leq p < \infty$. When $p = 2$, the space $W^{m,2}(\Omega) = H^m(\Omega)$ is a Hilbert space endowed with a scalar product and a norm. For example, for $m = 1$ the scalar product is

$$((u, v))_\Omega = (u, v)_\Omega + \sum_{i=1}^{d} (\partial_i u, \partial_i v)$$

and the norm is

$$\|u\|_{H^1(\Omega)} := ((u, u))_\Omega^{1/2}.$$

The d-dimensional vector functions with components in one of the previous spaces will be denoted by boldface letters, for example $\boldsymbol{L}^2(\Omega) = (L^2(\Omega))^d$.

Time discretization

In order to try to unify the notation we will introduce here the key concepts on time discretization to be used in this work. Considering a uniform partition of the time interval of size δt, and denoting by f^n an approximation to a time dependent function f at time $t^n = n\delta t$, for a parameter $\theta \in [0, 1]$, we will denote

$$f^{n+\theta} = \theta f^{n+1} + (1 - \theta) f^n,$$

$$\delta f^{n+1} = \delta^{(1)} f^{n+1} = f^{n+1} - f^n,$$

$$\delta^{(i+1)} f^{n+1} = \delta^{(i)} f^{n+1} - \delta^{(i)} f^n,\ i = 1, 2, 3, ...$$

Let us also define

$$D_t\left(\cdot\right) = \frac{\delta\left(\cdot\right)}{\delta t}.$$

The discrete operators $\delta^{(i+1)}$ are centered. We will also use backward difference operators

$$D_k f^{n+1} = \frac{1}{\gamma_k}\left(f^{n+1} - \sum_{i=0}^{k-1} \alpha_k^i\ f^{n-i}\right),$$

$$D_1 f^{n+1} = \delta f^{n+1} = f^{n+1} - f^n,$$

$$D_2 f^{n+1} = \frac{3}{2}\left(f^{n+1} - \frac{4}{3} f^n + \frac{1}{3} f^{n-1}\right),$$

as well as the backward extrapolation operators

$$\widetilde{f}_i^{n+1} = f^{n+1} - \delta^{(i)} f^{n+1} = f^{n+1} - \mathcal{O}\left(\delta t^i\right),$$

$$\widetilde{f}_1^{n+1} = f^n,$$

$$\widetilde{f}_2^{n+1} = 2f^n - f^{n-1}.$$

1.2 Two fluid Navier–Stokes equations

1.2.1 The (one fluid) Navier–Stokes equations

Before introducing the two fluid Incompressible Navier–Stokes equations, the typical one fluid version will be presented, as it the starting point from which the former are derived. The Navier–Stokes equations are the basic equations of fluid mechanics for incompressible flow and can be derived from the continuum mechanics conservation laws, see for example [8].

The Navier–Stokes equations, using an Eulerian description, for a fluid moving in the open domain Ω bounded by $\Gamma = \partial\Omega$ during the time interval (t_0, t_f) consist in finding a

velocity $\boldsymbol{u}$ and a pressure p such that

$$\rho\Big[\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\Big] - \nabla\cdot\boldsymbol{\sigma} = \boldsymbol{f} \qquad \text{in } \Omega\times(t_0,t_f), \tag{1.1}$$

$$\nabla\cdot\boldsymbol{u} = 0 \qquad \text{in } \Omega\times(t_0,t_f), \tag{1.2}$$

where ρ is the density, $\boldsymbol{\sigma}$ the stress tensor and $\boldsymbol{f}$ the vector external body forces, which includes the gravity force $\rho\boldsymbol{g}$ and buoyancy forces, if required. Using the constitutive equation for a Newtonian and isotropic fluid

$$\boldsymbol{\sigma} = -p\boldsymbol{I} + 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})$$

where μ is the dynamic viscosity, $\boldsymbol{I}$ is the identity tensor and $\boldsymbol{\varepsilon}(\cdot)$ the symmetric gradient operator, the momentum equation (1.1) can be rewritten in one of its usual forms

$$\rho\Big[\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\Big] - \nabla\cdot[2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})] + \nabla p = \boldsymbol{f} \qquad \text{in } \Omega\times(\mathrm{t}_0,\mathrm{t}_\mathrm{f}), \tag{1.3}$$

which we will call divergence form. For a constant μ and using the incompressibility constraint imposed by the continuity equation (1.2) the most usual form

$$\rho\Big[\partial_t \boldsymbol{u} + (\boldsymbol{u}\cdot\nabla)\boldsymbol{u}\Big] - \mu\Delta\boldsymbol{u} + \nabla p = \boldsymbol{f} \qquad \text{in } \Omega\times(\mathrm{t}_0,\mathrm{t}_\mathrm{f}),$$

which we will call Laplacian form, can be obtained.

Denoting by an over-bar prescribed values, the boundary conditions to be considered are:

$$\begin{aligned} \boldsymbol{u} &= \overline{\boldsymbol{u}} && \text{on } \Gamma_{\mathrm{du}}\times(t_0,t_f),\\ \boldsymbol{n}\cdot\boldsymbol{\sigma} &= \boldsymbol{t} && \text{on } \Gamma_{\mathrm{nu}}\times(t_0,t_f),\\ \boldsymbol{u}\cdot\boldsymbol{n} = 0, \quad \boldsymbol{n}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{g}_1 &= t_1, \quad \boldsymbol{n}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{g}_2 = t_2 && \text{on } \Gamma_{\mathrm{mu}}\times(t_0,t_f), \end{aligned} \tag{1.4}$$

where $\boldsymbol{n}$ is the unit outward normal to the boundary $\partial\Omega$ and vectors $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ (for the three-dimensional case) span the space tangent to Γ_{mu}. Observe that Γ_{du} is the part of the boundary with Dirichlet velocity conditions, Γ_{nu} the part with Neumann conditions (prescribed stress) and Γ_{mu} the part with mixed conditions. These three parts do not intersect and are a partition of the whole boundary $\partial\Omega$. Initial conditions

$$\boldsymbol{u} = \boldsymbol{u}_0 \quad \text{in } \Omega\times\{t_0\},$$

have to be appended to the problem.

In order to obtain the weak or variational formulation of the Navier–Stokes equations written in divergence form ((1.3) and (1.2)) we introduce the spaces

$$\boldsymbol{V}_0 \equiv \left\{ \boldsymbol{v} \in \boldsymbol{H}^1(\Omega) \mid \boldsymbol{v} = \boldsymbol{0} \text{ on } \Gamma_{\text{du}} \ , \ \boldsymbol{v} \cdot \boldsymbol{n} = 0 \text{ on } \Gamma_{\text{mu}} \right\},$$

$$\boldsymbol{V} \equiv \left\{ \boldsymbol{v} \in \boldsymbol{H}^1(\Omega) \mid \boldsymbol{v} = \overline{\boldsymbol{u}} \text{ on } \Gamma_{\text{du}} \ , \ \boldsymbol{v} \cdot \boldsymbol{n} = 0 \text{ on } \Gamma_{\text{mu}} \right\},$$

$$\boldsymbol{V}_t \equiv L^2(t_0, t_f; \boldsymbol{V}),$$

$$Q \equiv \begin{cases} L^2(\Omega) & \text{if } \Gamma_{\text{nu}} \neq \emptyset \\ L^2(\Omega)/\mathbb{R} & \text{if } \Gamma_{\text{nu}} = \emptyset \end{cases}$$

$$Q_t \equiv L^1(t_0, t_f; Q)$$

The weak form is then obtained by multiplying each of the momentum equations (1.3) by an arbitrary element of $\boldsymbol{V}_0$, $\boldsymbol{v}$, and the continuity equation (1.2) by an arbitrary element of Q, q, and integrating the term corresponding to the stress tensor by parts.

The weak form of problem (1.3, 1.2) with the boundary conditions we have just defined is: Find $\boldsymbol{u} \in \boldsymbol{V}_t$, $p \in Q_t$ such that

$$\rho \int_\Omega \partial_t \boldsymbol{u} \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \rho \int_\Omega \left[(\boldsymbol{u} \cdot \nabla) \boldsymbol{u} \right] \cdot \boldsymbol{v} \ \mathrm{d}\Omega + 2 \int_\Omega \mu \boldsymbol{\varepsilon}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \ \mathrm{d}\Omega$$
$$- \int_\Omega p \ \nabla \cdot \boldsymbol{v} \ \mathrm{d}\Omega = \rho \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_{\Gamma_{\text{nu}}} \boldsymbol{t} \cdot \boldsymbol{v} \ \mathrm{d}\Gamma + \int_{\Gamma_{\text{mu}}} (t_1 \boldsymbol{g}_1 + t_2 \boldsymbol{g}_2) \cdot \boldsymbol{v} \ \mathrm{d}\Gamma$$

$$\int_\Omega q \ \nabla \cdot \boldsymbol{u} \ \mathrm{d}\Omega = 0$$

for all $(\boldsymbol{v}, q) \in \boldsymbol{V}_0 \times Q$.

1.2.2 The two fluid Navier–Stokes equations

The two fluid Navier–Stokes equations on a domain $\Omega = \Omega_1 \cup \Omega_2$ separated by a moving interface Γ_{int} can be obtained starting from the Navier–Stokes equations defined on each domain [18] , written in divergence form

$$\rho_1 \Big[\partial_t \boldsymbol{u}_1 + (\boldsymbol{u}_1 \cdot \nabla) \boldsymbol{u}_1 \Big] - \nabla \cdot \boldsymbol{\sigma}_1 = \boldsymbol{f}_1 \qquad \text{in } \Omega_1 \times (t_0, t_f),$$

$$\nabla \cdot \boldsymbol{u}_1 = 0 \qquad \text{in } \Omega_1 \times (t_0, t_f),$$

and

$$\rho_2 \Big[\partial_t \boldsymbol{u}_2 + (\boldsymbol{u}_2 \cdot \nabla)\boldsymbol{u}_2 \Big] - \nabla \cdot \boldsymbol{\sigma}_2 = \boldsymbol{f}_2 \qquad \text{in } \Omega_2 \times (t_0, t_f),$$

$$\nabla \cdot \boldsymbol{u}_2 = 0 \qquad \text{in } \Omega_2 \times (t_0, t_f),$$

In order to simplify the presentation we will suppose that the only Neumann boundary in both Ω_1 and Ω_2 corresponds to the interface and that $\Gamma_{\text{mu}} = \emptyset$. The boundary conditions at the interface are obtained as follows. Since the flow is viscous

$$\boldsymbol{u}_1 = \boldsymbol{u}_2 \qquad \text{on } \Gamma_{\text{int}}.$$

On the other hand, the balance of surface forces on the interface gives

$$(\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2) \cdot \boldsymbol{n} = k \, \varkappa \, \boldsymbol{n} \qquad \text{on } \Gamma_{\text{int}} \tag{1.5}$$

where the term on the right hand side models the surface tension; k is a constant coefficient that depends on the two fluids in contact (usually σ is used in the literature but we have used k to avoid confusions), $\varkappa$ is the local curvature of the interface and $\boldsymbol{n}$ is the normal pointing towards the positive curvature region. In mould filling simulations the effects of surface tension are usually negligible and therefore will not be taken into account in this thesis ($k = 0$). Nevertheless they will be included in the derivation of the two fluid Navier–Stokes equations. Using the same functional spaces as in the one fluid case, the momentum equations corresponding to each of the two fluids are multiplied by an arbitrary element of $\boldsymbol{V}_0$, $\boldsymbol{v}$, and integrated over their corresponding domains, the term corresponding to the stress tensor is integrated by parts, and the variational formulations corresponding to each of the two fluids are added. The same procedure is followed for the continuity equation using an arbitrary element of Q, q. Finally, defining

$$\boldsymbol{u},\ p,\ \rho,\ \mu, \boldsymbol{f}, \boldsymbol{\sigma} = \begin{cases} \boldsymbol{u}_1\ p_1,\ \rho_1,\ \mu_1,\ \boldsymbol{f}_1,\ \boldsymbol{\sigma}_1 & \boldsymbol{x} \in \Omega_1, \\ \boldsymbol{u}_2\ p_2,\ \rho_2,\ \mu_2,\ \boldsymbol{f}_2,\ \boldsymbol{\sigma}_2 & \boldsymbol{x} \in \Omega_2, \end{cases} \tag{1.6}$$

the unified variational formulation is obtained. Find $\boldsymbol{u} \in \boldsymbol{V}_t$, $p \in Q_t$ such that

$$\begin{aligned} &\int_\Omega \partial_t \rho \boldsymbol{u} \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_\Omega \rho \left[(\boldsymbol{u} \cdot \nabla) \boldsymbol{u} \right] \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \ \mathrm{d}\Omega \\ &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_{\Gamma_{\text{int}}} \left[(\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2) \cdot \boldsymbol{n} \right] \cdot \boldsymbol{v} \ \mathrm{d}\Gamma \qquad \forall \ \boldsymbol{v} \in \boldsymbol{V}_0 \end{aligned} \tag{1.7}$$

$$\int_\Omega q \ \nabla \cdot \boldsymbol{u} \ \mathrm{d}\Omega = 0 \qquad \forall \ q \in Q.$$

Using the equation for the interfacial forces, (1.5), we finally obtain

$$\begin{aligned} &\int_\Omega \partial_t \rho \boldsymbol{u} \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_\Omega \rho \left[(\boldsymbol{u} \cdot \nabla) \boldsymbol{u} \right] \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_\Omega \boldsymbol{\sigma}(\boldsymbol{u}) : \boldsymbol{\varepsilon}(\boldsymbol{v}) \ \mathrm{d}\Omega \\ &= \int_\Omega \boldsymbol{f} \cdot \boldsymbol{v} \ \mathrm{d}\Omega + \int_{\Gamma_{\text{int}}} k \ \varkappa \ \boldsymbol{n} \cdot \boldsymbol{v} \ \mathrm{d}\Gamma \qquad \forall \ \boldsymbol{v} \in \boldsymbol{V}_0 \end{aligned}$$

$$\int_\Omega q \ \nabla \cdot \boldsymbol{u} \ \mathrm{d}\Omega = 0 \qquad \forall \ q \in Q.$$

In equation (1.7) we have obtained the term

$$\int_{\Gamma_{\text{int}}} \left[(\boldsymbol{\sigma}_1 - \boldsymbol{\sigma}_2) \cdot \boldsymbol{n} \right] \cdot \boldsymbol{v} \ \mathrm{d}\Gamma$$

because we started from the divergence form of the equations. This is desirable since, as we have already said, its value is given by (1.5). If the Laplacian form had been used, the integral on the interface would have been replaced by

$$\int_{\Gamma_{\text{int}}} \{ [(-p_1 \boldsymbol{I} + \mu_1 \nabla \boldsymbol{u}_1) - (-p_2 \boldsymbol{I} + \mu_2 \nabla \boldsymbol{u}_2)] \cdot \boldsymbol{n} \} \cdot \boldsymbol{v} \ \mathrm{d}\Gamma$$

which is not related to the balance of surface forces on the interface. Therefore the divergence form will always be used in this thesis, unless otherwise indicated.

Before continuing with the discretization of the equations some dimensionless numbers can be presented. The Froude number represents the relation between the inertial and gravitational forces and is defined by

$$Fr = \frac{U^2}{gL}$$

where g is the gravity acceleration, U is a characteristic velocity and L a characteristic length. Two Reynolds numbers can be defined

$$Re_1 = \frac{UL\rho_1}{\mu_1}$$

$$Re_2 = \frac{UL\rho_1}{\mu_2}.$$

If surface tension is taken into account, the Weber number is defined as

$$We = \frac{U^2 L\rho_1}{k}$$

where k has been defined in (1.5).

1.2.3 Basic discretized problem

In this subsection we will introduce the space and time discretization of the weak two fluid NS equations. Also the linearization of the convective term will be described.

The linearization of the convective term can be performed at the continuous or variational level indistinctly. The approximation we will use is well known and reads as follows:

$$\left[\left(\boldsymbol{u}\cdot\nabla\right)\boldsymbol{u}\right]^{i+1} \approx \left(\boldsymbol{u}^i\cdot\nabla\right)\boldsymbol{u}^{i+1} + \beta\left(\boldsymbol{u}^{i+1}\cdot\nabla\right)\boldsymbol{u}^i - \beta\left(\boldsymbol{u}^i\cdot\nabla\right)\boldsymbol{u}^i$$

where i is the iteration counter and β can be zero or one. When $\beta = 0$ the method is known as Picard linearization and when $\beta = 1$ it is known as Newton-Raphson linearization. The former is simpler and has the advantage that it can be shown to converge linearly if the convection is not too high. For the latter, a quadratic convergence can be proved but only if the initial guess is close enough to the exact solution [21]. Therefore, the typical numerical strategy is to first solve some Picard iterations to take advantage of its robustness and then to switch to the Newton-Raphson method for improved convergence.

Regarding the time discretization of problem (1.7) two options will be presented. The first one is the generalized trapezoidal rule, which gives place to the following problem: given $\boldsymbol{u}^n$, find $\boldsymbol{u}^{n+1} \in \boldsymbol{V}$ and $p^{n+1} \in Q$ such that

$$\begin{aligned}
&\int_\Omega \rho^{n+\theta}\,\frac{\boldsymbol{u}^{n+\theta}-\boldsymbol{u}^n}{\theta\delta t}\cdot\boldsymbol{v}\;\mathrm{d}\Omega + \int_\Omega \rho^{n+\theta}\left[\left(\boldsymbol{u}^{n+\theta}\cdot\nabla\right)\boldsymbol{u}^{n+\theta}\right]\cdot\boldsymbol{v}\;\mathrm{d}\Omega \\
&\qquad +2\int_\Omega \mu^{n+\theta}\boldsymbol{\varepsilon}\left(\boldsymbol{u}^{n+\theta}\right):\boldsymbol{\varepsilon}\left(\boldsymbol{v}\right)\;\mathrm{d}\Omega - \int_\Omega p^{n+\theta}\,\nabla\cdot\boldsymbol{v}\;\mathrm{d}\Omega \\
&= \int_\Omega \boldsymbol{f}^{n+\theta}\cdot\boldsymbol{v}\;\mathrm{d}\Omega + \int_{\Gamma_{\mathrm{nu}}} \boldsymbol{t}^{n+\theta}\cdot\boldsymbol{v}\;\mathrm{d}\Gamma + \int_{\Gamma_{\mathrm{nu}}} \left(t_1^{n+\theta}\boldsymbol{g}_1 + t_2^{n+\theta}\boldsymbol{g}_2\right)\cdot\boldsymbol{v}\;\mathrm{d}\Gamma
\end{aligned}$$

$$\int_\Omega q \, \nabla \cdot \boldsymbol{u}^{n+\theta} \, \mathrm{d}\Omega = 0$$

for all $(\boldsymbol{v}, q) \in \boldsymbol{V}_0 \times Q$. Once the algorithm has produced a solution at $t^{n+\theta}$, the velocity field at t^{n+1} can be updated from the velocity at $t^{n+\theta}$ by using the relation $\boldsymbol{u}^{n+1} = [\boldsymbol{u}^{n+\theta} - (1-\theta)\boldsymbol{u}^n]/\theta$. The force term $\boldsymbol{f}^{n+\theta}$ in the momentum equation has to be understood as the time average in the interval $[t^n, t^{n+1}]$, even though we use the superscript $n+\theta$ to characterize it. The same applies for $\boldsymbol{t}^{n+\theta}$, $t_1^{n+\theta}$ and $t_2^{n+\theta}$. The pressure value has been identified as the pressure at $t^{n+\theta}$, although this is irrelevant for the velocity approximation. The values of interest of θ are $\theta = 1/2$, that corresponds to the second order Crank-Nicolson scheme and $\theta = 1$, that corresponds to the backward Euler method.

The second option is to use backward differencing (BDF) time integration schemes using the discrete operators defined in Section 1.1. The time discretized problem then reads: given $\boldsymbol{u}^n$, find $\boldsymbol{u}^{n+1} \in \boldsymbol{V}$ and $p^{n+1} \in Q$ such that

$$\begin{aligned}
&\int_\Omega \frac{\rho^{n+1}}{\delta t} D_k \boldsymbol{u}^{n+1} \cdot \boldsymbol{v} \, \mathrm{d}\Omega + \int_\Omega \rho^{n+1} \left[\left(\boldsymbol{u}^{n+1} \cdot \nabla \right) \boldsymbol{u}^{n+1} \right] \cdot \boldsymbol{v} \, \mathrm{d}\Omega \\
&\qquad +2 \int_\Omega \mu^{n+1} \boldsymbol{\varepsilon}\left(\boldsymbol{u}^{n+1} \right) : \boldsymbol{\varepsilon}\left(\boldsymbol{v} \right) \, \mathrm{d}\Omega - \int_\Omega p^{n+1} \, \nabla \cdot \boldsymbol{v} \, \mathrm{d}\Omega \\
= &\int_\Omega \boldsymbol{f}^{n+1} \cdot \boldsymbol{v} \, \mathrm{d}\Omega + \int_{\Gamma_{\mathrm{nu}}} \boldsymbol{t}^{n+1} \cdot \boldsymbol{v} \, \mathrm{d}\Gamma + \int_{\Gamma_{\mathrm{nu}}} \left(t_1^{n+1} \boldsymbol{g}_1 + t_2^{n+1} \boldsymbol{g}_2 \right) \cdot \boldsymbol{v} \, \mathrm{d}\Gamma
\end{aligned}$$

$$\int_\Omega q \, \nabla \cdot \boldsymbol{u}^{n+1} \, \mathrm{d}\Omega = 0$$

for all $(\boldsymbol{v}, q) \in \boldsymbol{V}_0 \times Q$. The first order versions of both methods coincide. For the second order time discretizations the benefits of each of the methods are subtle for the one fluid NS equations. For the two fluid case we prefer the BDF scheme. Since the fluid properties (ρ and μ) at a given point vary in time, as defined in (1.6), it is much better to use the properties at time t^{n+1} which can be obtained from the level set function (whose value is known at t^{n+1}) than those at time $t^{n+\theta}$ which need to be somehow approximated.

The final ingredient for obtaining the basic (without stabilization) discretized problem is the space discretization, that we build with the finite element method (see for example [60] or [66]). The key step is to construct the discrete linear subspaces $\boldsymbol{V}_h \subset \boldsymbol{V}$, $\boldsymbol{V}_{0h} \subset \boldsymbol{V}_0$ and $Q_h \subset Q$ that approximate the continuous spaces. Let $\boldsymbol{V}_h^*$ and Q_h^* be the finite element

spaces to interpolate vector and scalar functions, respectively, constructed in the usual manner from a finite element partition $\Omega = \bigcup \Omega^e$, $e = 1, ..., n_{\text{el}}$, where n_{el} is the number of elements. In this thesis the same interpolation will be used for both the velocity and the pressure, except in Chapter 2 where the pressure space will be enriched. In particular P1-P1 interpolations (continuous and linear in both velocity and pressure) will be preferred. From spaces $\boldsymbol{V}_h^*$ and Q_h^* one can construct the subsets $\boldsymbol{V}_{h,u}$ and Q_h for the velocity and the pressure, respectively. The former incorporates the Dirichlet conditions for the velocity components (and also the mixed conditions corresponding to the normal velocity) and the latter has one pressure fixed to zero if the normal component of the velocity is prescribed on the whole boundary. The space of velocity test functions, denoted by $\boldsymbol{V}_h$, is constructed as $\boldsymbol{V}_{h,u}$ but with functions vanishing on the Dirichlet boundary.

The monolithic discrete problem associated with the Navier–Stokes equations, discretizing in time using a BDF scheme, and linearizing the convective term using a Picard scheme (in order to simply the presentation), can be written as follows: Given a velocity $\boldsymbol{u}_h^n$ at time t^n and a guess for the unknowns at an iteration $i-1$ at time t^{n+1}, find $\boldsymbol{u}_h^{n+1,i} \in \boldsymbol{V}_h$ and $p_h^{n+1,i} \in Q_h$, by solving the discrete variational problem:

$$\begin{aligned}
&\int_\Omega \frac{\rho}{\delta t} D_k \boldsymbol{u}_h^{n+1} \cdot \boldsymbol{v}_h \, \mathrm{d}\Omega + \int_\Omega \rho(\boldsymbol{u}_h^{n+1,i-1} \cdot \nabla)\boldsymbol{u}_h^{n+1,i} \cdot \boldsymbol{v}_h \, \mathrm{d}\Omega \\
&\quad + \int_\Omega \mu \boldsymbol{\varepsilon}(\boldsymbol{u}_h^{n+1,i}) : \boldsymbol{\varepsilon}(\boldsymbol{v}_h) \, \mathrm{d}\Omega - \int_\Omega \nabla \cdot \boldsymbol{v}_h p_h^{n+1,i} \, \mathrm{d}\Omega - \int_\Omega \boldsymbol{v}_h \cdot \boldsymbol{f} \, \mathrm{d}\Omega \\
&\quad - \int_{\Gamma_{\text{nu}}} \boldsymbol{t}^{n+1} \cdot \boldsymbol{v}_h \, \mathrm{d}\Gamma - \int_{\Gamma_{\text{nu}}} \left(t_1^{n+1}\boldsymbol{g}_1 + t_2^{n+1}\boldsymbol{g}_2\right) \cdot \boldsymbol{v}_h \, \mathrm{d}\Gamma = 0,
\end{aligned}$$

$$\int_\Omega q_h \nabla \cdot \boldsymbol{u}_h^{n+1,i} \, \mathrm{d}\Omega = 0 \ ,$$

for $i = 1, 2, ...$ until convergence, that is to say, until $\boldsymbol{u}_h^{n+1,i-1} \approx \boldsymbol{u}_h^{n+1,i}$ and $p_h^{n+1,i} \approx p_h^{n+1,i-1}$ in the norm defined by the user. In order to simplify the notation we use $\rho \equiv \rho^{n+1}$ and $\mu \equiv \mu^{n+1}$.

The enrichment technique presented in Chapter 2 can be understood as a modification of the pressure space Q_h to $\hat{Q}_h$, with $Q_h \subset \hat{Q}_h$. Apart from this, the resulting formulation follows exactly the previous setting.

1.2.4 Stabilized problem

The discretized problem presented in the previous Subsection needs to be stabilized before it can be solved numerically for two well known reasons. The first one is related to the instabilities that appear in convection-dominated flows using reasonably sized meshes. The second one is related to the use velocity and pressure finite element spaces that do not satisfy the div-stability restriction (inf-sup condition) [12] , as is the case of equal interpolation for both unknowns that we use in this thesis.

A wide range of stabilization techniques can be found in the literature, among them we can mention, using their commonly used acronyms : SUPG [14], PSPG [108], GLS [62], CBS [34,119], FIC [86], ASGS [24] and OSS [23,25]. In this work two of them will be used: the Algebraic version of the Subgrid Scale stabilization method, referred to as ASGS [24] and the Orthogonal Sub-scale stabilization method, referred to as OSS [23, 25]. In a recent article [28] we have compared numerically the two methods we will use in this work with the Characteristic-Based-Split (CBS) stabilization technique for the incompressible Navier–Stokes equations.

The two Subgrid Scale (SGS) formulations we present deal with convection and pressure stabilization using the same approach. The idea of SGS methods was proposed in [61], although it is inherent in other numerical formulations. The key idea is to approximate $\boldsymbol{u} \approx \boldsymbol{u}_h + \widetilde{\boldsymbol{u}}$ and $p \approx p_h$, that is, the velocity is approximated by its finite element component plus an additional term that is called *subgrid scale* or *subscale*.

We call $\boldsymbol{u}^{n+1} \approx \boldsymbol{u}_*^{n+1} := \boldsymbol{u}_h^{n+1} + \widetilde{\boldsymbol{u}}^{n+1}$and $p^{n+1} \approx p_h^{n+1}$ the velocity and the pressure at t^{n+1}. As previously mentioned, the spatial interpolation for $\boldsymbol{u}_h^{n+1}$and $p_h^{n+1}$are constructed using the standard finite element interpolation. In particular, equal velocity-pressure interpolation is possible.

The important point is the behavior assumed for $\widetilde{\boldsymbol{u}}^{n+1}$. It is assumed that it vanishes on the interelement boundaries, that is, it is a bubble-like function [7, 13]. However, contrary to what is commonly done, we do not assume any particular behavior of $\widetilde{\boldsymbol{u}}^{n+1}$ within the element domains. We will show later on how to approximate it.

If in the space continuous and time discrete problem $\boldsymbol{u}$ is replaced by $\boldsymbol{u}_*^{n+1} := \boldsymbol{u}_h^{n+1} + \widetilde{\boldsymbol{u}}^{n+1}$, p is replaced by p_h^{n+1}, the terms involving $\widetilde{\boldsymbol{u}}_h^{n+1}$ are integrated by parts, and the test functions are taken in the finite element space, one gets

$$\delta t \int_\Omega \left[\mu \boldsymbol{\nabla} \boldsymbol{u}_h^{n+1} : \boldsymbol{\nabla} \boldsymbol{v}_h + \rho \left(\boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{u}_h^{n+1}\right) \cdot \boldsymbol{v}_h - p_h^{n+1} \nabla \cdot \boldsymbol{v}_h + q_h \nabla \cdot \boldsymbol{u}_h^{n+1} - \boldsymbol{f}^{n+1} \cdot \boldsymbol{v}_h\right] \, \mathrm{d}\Omega$$
$$+ \int_\Omega \rho \left[\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n\right] \cdot \boldsymbol{v}_h \, \mathrm{d}\Omega - \delta t \sum_e \int_{\Omega^e} \widetilde{\boldsymbol{u}}^{n+1} \left[\mu \boldsymbol{\Delta}_h \boldsymbol{v}_h + \rho \boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{v}_h + \nabla q_h\right] \, \mathrm{d}\Omega = 0, \tag{1.8}$$

where for simplicity we have used a first order scheme and the Laplacian form and $\boldsymbol{u} = \boldsymbol{0}$ on $\partial\Omega$. The notation $\boldsymbol{\Delta}_h$ is used to indicate that the Laplacian needs to be evaluated element by element. Equation (1.8) must hold for all test functions $\boldsymbol{v}_h$ and q_h in their corresponding finite element spaces.

The equation for the subscales $\widetilde{\boldsymbol{u}}^{n+1}$ is obtained by taking the velocity test function in its space and $q = 0$. The next step is to model the resulting equation. The first possibility, which gives rise to the ASGS method [24], is to take

$$\widetilde{\boldsymbol{u}}^{n+1} = -\tau_1 \boldsymbol{R}_h^{n+1},$$

where τ_1 is a numerical parameter and $\boldsymbol{R}_h^{n+1}$is the residual defined as:

$$\boldsymbol{R}_h^{n+1} = \rho \boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{u}_h^{n+1} - \mu \boldsymbol{\Delta}_h \boldsymbol{u}_h^{n+1} + \nabla p_h^{n+1} - \boldsymbol{f}^{n+1} + \rho \frac{\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n}{\delta t}.$$

The second option, which gives rise to the OSS method [23,25], is to impose the subscales to be orthogonal to the finite element space,

$$\widetilde{\boldsymbol{u}}^{n+1} = -\tau_1 P_h^\perp \boldsymbol{R}_h^{n+1} = -\tau_1 (\boldsymbol{R}_h^{n+1} - P_h(\boldsymbol{R}_h^{n+1})),$$

where P_h is the projection onto the finite element space. The advantage of this approach is discussed in [25]. From the accuracy point of view, it is less diffusive that the ASGS approach and yields better resolution of sharp gradients of the unknowns.

In the previous approximations for $\widetilde{\boldsymbol{u}}^{n+1}$, the temporal variation of the subscales has been considered negligible, in [25] they are called quasi-static subscales. It is the option

commonly used in the literature. If temporal variation of the subscales would not be neglected they would need to be tracked. This promising approach has been proposed in [25], but little has been done up to the moment. A recent article can be found in [30].

With all the approximations introduced heretofore, the final discrete problem to be solved for $\boldsymbol{u}_h^{n+1}$and p_h^{n+1} using the ASGS method is

$$\int_\Omega \Big[\frac{\rho}{\delta t}(\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n)\cdot\boldsymbol{v}_h+\mu\nabla\boldsymbol{u}_h^{n+1}:\nabla\boldsymbol{v}_h+\rho(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1})\cdot\boldsymbol{v}_h-p_h^{n+1}\ \nabla\cdot\boldsymbol{v}_h$$
$$-\boldsymbol{f}^{n+1}\cdot\boldsymbol{v}_h\Big]\,\mathrm{d}\Omega+\sum_e\int_{\Omega^e}\tau_1(\mu\Delta_h\boldsymbol{v}_h+\rho\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{v}_h)\cdot\boldsymbol{R}_h^{n+1}\,\mathrm{d}\Omega=0,$$

$$\int_\Omega\Big[q_h\nabla\cdot\boldsymbol{u}_h^{n+1}\Big]\,\mathrm{d}\Omega+\sum_e\int_{\Omega^e}\tau_1\nabla q_h\cdot\boldsymbol{R}_h^{n+1}\,\mathrm{d}\Omega=0.$$

In the OSS case a preliminary version can be obtained by replacing $\boldsymbol{R}_h^{n+1}$ with $P_h^\perp\boldsymbol{R}_h^{n+1}$ to obtain,

$$\int_\Omega \Big[\frac{\rho}{\delta t}(\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n)\cdot\boldsymbol{v}_h+\mu\nabla\boldsymbol{u}_h^{n+1}:\nabla\boldsymbol{v}_h+\rho(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1})\cdot\boldsymbol{v}_h-p_h^{n+1}\ \nabla\cdot\boldsymbol{v}_h$$
$$-\boldsymbol{f}^{n+1}\cdot\boldsymbol{v}_h\Big]\,\mathrm{d}\Omega+\sum_e\int_{\Omega^e}\tau_1(\mu\Delta_h\boldsymbol{v}_h+\rho\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{v}_h)\cdot P_h^\perp\boldsymbol{R}_h^{n+1}\,\mathrm{d}\Omega=0,$$

$$\int_\Omega\Big[q_h\nabla\cdot\boldsymbol{u}_h^{n+1}\Big]\,\mathrm{d}\Omega+\sum_e\int_{\Omega^e}\tau_1\nabla q_h\cdot P_h^\perp\boldsymbol{R}_h^{n+1}\,\mathrm{d}\Omega=0.$$

In the previous equation some terms turn out to be zero and others are neglected before the final version presented in [25] is obtained. Actually in [25] only the constant density case has been analyzed. For the two fluid case we have adapted the equations from [32]. When the previous OSS formulation was tested on two phase flow problems, where the density can vary in three orders of magnitude, much poorer results than with the ASGS formulation were obtained. Detailed inspection of the problem showed that the residual on integration points on opposite side of the interface varied roughly proportionately to the density. Then it became obvious that the projection of a residual with such variations might be the source of the errors we were observing. The solution we adopted was to use a modified projection

$$P_{h\rho}(\boldsymbol{R}_h^{n+1})=\rho P_h(\frac{\boldsymbol{R}_h^{n+1}}{\rho}).$$

This has been a key element in the successful solution of different density flows using the OSS formulation.

From the theoretical point of view, the use of $P_{h\rho}(\boldsymbol{R}_h^{n+1})$ can be related to the fact that the L^2 projection P_h is introduced in [25] as an approximation to a τ_1 weighted projection where for the variable density case τ_1 can be defined at element level as

$$\tau_1 = \left[\rho\left(\frac{4\upsilon}{(h^e)^2} + \frac{2|\boldsymbol{u}^e|}{h^e}\right)\right]^{-1}$$

where h^e and $|\boldsymbol{u}^e|$ are a typical length and a velocity norm of element e, respectively. It now becomes obvious that a ρ^{-1} weighted projection might be a better approximation to the τ_1 weighted projection than a straightforward L^2 projection.

We now proceed to obtain the final version of the of the OSS stabilized problem for the variable density flows. The transient term in $\frac{\boldsymbol{R}_h^{n+1}}{\rho}$ belongs to the finite element space and therefore its orthogonal projection is zero. Note that for variable density flows this does not happen if unweighted projection, $P_h(\boldsymbol{R}_h^{n+1})$, is used. Regarding the force term, in most cases (but not in variable density flows under gravity forces) it belongs to the finite element space and therefore its orthogonal projection is zero. In other cases it can be neglected because it introduces an error of the same order as the optimal error that can be expected [25]. In our formulation we have nevertheless conserved this term in the residual to be used in the stabilization. For the low Froude number flows that we shall be interested in our mould filling simulations the two most important terms are the force term and the pressure gradient. In the limiting case of flows at rest they balance each other. Therefore, for the cases we are interested in, it seems much more logical to preserve the force term.

As explained in [25], second order derivatives of finite element functions within element interiors can be neglected and the consistency of the OSS method can be preserved. For linear elements these terms are zero. Instead, for the ASGS stabilization, if the viscous terms are neglected consistency is lost. Despite the ASGS method is consistent, its implementation for linear elements, is identical to the implementation of a non consistent scheme. We can conclude that the OSS scheme is much better suited for linear elements

than the ASGS scheme. In [65] an improvement, that uses an L^2 projection for the diffusive term in the residual, is introduced to mitigate the weakness of the ASGS method for linear elements.

Taking into account the previous comments, the final discrete problem to be solved for $\boldsymbol{u}_h^{n+1}$and p_h^{n+1} using the OSS method is

$$\int_\Omega \Big[\frac{\rho}{\delta t}(\boldsymbol{u}_h^{n+1}-\boldsymbol{u}_h^n)\cdot\boldsymbol{v}_h+\mu\nabla\boldsymbol{u}_h^{n+1}:\nabla\boldsymbol{v}_h+\rho(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1})\cdot\boldsymbol{v}_h-p_h^{n+1}\nabla\cdot\boldsymbol{v}_h-\boldsymbol{f}^{n+1}\cdot\boldsymbol{v}_h\Big]\,\mathrm{d}\Omega$$
$$+\sum_e\int_{\Omega^e}\tau_1\rho\,(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{v}_h)\cdot\Big[\big(\rho\,\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}+\nabla p_h^{n+1}-\boldsymbol{f}^{n+1}\big)$$
$$-\rho P_h\Big(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}+\frac{\nabla p_h^{n+1}}{\rho}-\frac{\boldsymbol{f}^{n+1}}{\rho}\Big)\Big]\,\mathrm{d}\Omega=0,$$

$$\int_\Omega\Big[q_h\nabla\cdot\boldsymbol{u}_h^{n+1}\Big]\,\mathrm{d}\Omega$$
$$+\sum_e\int_{\Omega^e}\tau_1\nabla q_h\cdot\Big[\big(\rho\,\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}+\nabla p_h^{n+1}-\boldsymbol{f}^{n+1}\big)$$
$$-\rho P_h\Big(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}+\frac{\nabla p_h^{n+1}}{\rho}-\frac{\boldsymbol{f}^{n+1}}{\rho}\Big)\Big]\,\mathrm{d}\Omega=0.$$

The key terms for the stabilization of the convective and pressure terms are $\tau_1\rho\ \big(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{v}_h\big)\cdot\rho\ \big(\boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}\big)$ and $\tau_1\nabla q_h\cdot\nabla p_h^{n+1}$ respectively. These terms appear in both formulations we have presented but also in most other stabilization techniques.

The reasons for having worked with two stabilization techniques are both practical and theoretical. On the practical side we have worked with the ASGS technique because it was the technique that was implemented in the monolithic version of our code. For the segregation methods, that will be presented in Chapter 5, our code only had a preliminary version that used the OSS method. Another reason for using ASGS is that it is a more widely used technique than OSS. Actually not ASGS on its own, but if one also counts GLS that is very similar to ASGS. As has already been mentioned, some of the advantages and disadvantages of OSS have been discussed in [25] but up to now there is no clear favorite method. As we have already mentioned, for P1-P1 elements OSS seems a better choice.

The OSS stabilization can be reformulated so that instead of working with the orthogonal projection of the convective and pressure terms together, two separate

projections can be used [25]. We will call this version the split OSS. As in the non split version, for the variable density case, we shall work with the projections weighted by $\frac{1}{\rho}$. Moreover, for the low Froude number flows instead of working with ∇p_h^{n+1}, as is usually done, we shall work with $\nabla p_h^{n+1} - \boldsymbol{f}^{n+1}$ for the same reasons we have explained for the non split version.

The split OSS version we use reads

$$\int_\Omega \Big[\frac{\rho}{\delta t}(\boldsymbol{u}_h^{n+1} - \boldsymbol{u}_h^n) \cdot \boldsymbol{v}_h + \mu \nabla \boldsymbol{u}_h^{n+1} : \nabla \boldsymbol{v}_h + \rho(\boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{u}_h^{n+1}) \cdot \boldsymbol{v}_h - p_h^{n+1} \nabla \cdot \boldsymbol{v}_h - \boldsymbol{f}^{n+1} \cdot \boldsymbol{v}_h \Big] \, \mathrm{d}\Omega + \sum_e \int_{\Omega^e} \tau_1 \rho \, (\boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{v}_h) \cdot \left[\left(\rho \, \boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{u}_h^{n+1} \right) - \rho P_h \left(\, \boldsymbol{u}_h^{n+1} \cdot \nabla \boldsymbol{u}_h^{n+1} \right) \right] \, \mathrm{d}\Omega = 0, \tag{1.9}$$

$$\int_\Omega \Big[q_h \nabla \cdot \boldsymbol{u}_h^{n+1} \Big] \, d\Omega + \sum_e \int_{\Omega^e} \tau_1 \nabla q_h \cdot \left[\left(\nabla p_h^{n+1} - \boldsymbol{f}^{n+1} \right) - \rho P_h \left(\frac{\nabla p_h^{n+1}}{\rho} - \frac{\boldsymbol{f}^{n+1}}{\rho} \right) \right] \, \mathrm{d}\Omega = 0.$$

In some situations the introduction of a pressure subscale $\widetilde{p}^{n+1}$ can also be advantageous [25] because it helps to enforce the incompressibility of the flow that can be excessively relaxed when only the velocity subscale is introduced. It is approximated as

$$\widetilde{p}^{n+1} = -\tau_2 \left[\nabla \cdot \boldsymbol{u}_h^{n+1} - \xi P_h \left(\nabla \cdot \boldsymbol{u}_h^{n+1} \right) \right],$$

with $\xi = 1$ in the OSS case and zero otherwise. For the OSS method this term would control $\nabla \cdot \boldsymbol{u}$ in the space orthogonal to the finite element space, but since we want this to happen in the whole space we have used $\xi = 0$ for both OSS and ASGS cases.

As a summary to the ideas presented up to now we rewrite the monolithic divergence form Navier-Stokes equations using Picard linearization, BDF time discretization and ASGS stabilization. Given a velocity $\boldsymbol{u}_h^n$ at time t^n (and also at previous times as required by D_k) and a guess for the unknowns at an iteration $i-1$ at time t^{n+1}, find $\boldsymbol{u}_h^{n+1,i} \in \boldsymbol{V}_h$

and $p_h^{n+1,i} \in Q_h$, by solving the discrete variational problem:

$$\begin{aligned}
&\int_\Omega \frac{\rho}{\delta t} D_k \boldsymbol{u}_h^{n+1} \cdot \boldsymbol{v}_h \, d\Omega + \int_\Omega \rho(\boldsymbol{u}_h^{n+1,i-1} \cdot \nabla)\boldsymbol{u}_h^{n+1,i} \cdot \boldsymbol{v}_h \, \mathrm{d}\Omega \\
&\quad + \int_\Omega \mu \boldsymbol{\varepsilon}(\boldsymbol{u}_h^{n+1,i}) : \boldsymbol{\varepsilon}(\boldsymbol{v}_h) \, \mathrm{d}\Omega - \int_\Omega \nabla \cdot \boldsymbol{v}_h p_h^{n+1,i} \, \mathrm{d}\Omega - \int_\Omega \boldsymbol{v}_h \cdot \boldsymbol{f} \, \mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_1^{n+1,i-1} \Big[\mu \, \Delta \boldsymbol{v}_h + \rho \, (\boldsymbol{u}_h^{n+1,i-1} \cdot \nabla)\boldsymbol{v}_h \Big] \cdot \Big[\frac{\rho}{\delta t} D_k \boldsymbol{u}_h^{n+1} \\
&\qquad - \mu \Delta \boldsymbol{u}_h^{n+1,i} + \rho \, (\boldsymbol{u}_h^{n+1,i-1} \cdot \nabla)\boldsymbol{u}_h^{n+1,i} + \nabla p_h^{n+1,i} - \boldsymbol{f} \Big] \, \mathrm{d}\Omega \\
&\qquad + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_2^{n+1,i-1} (\nabla \cdot \boldsymbol{v}_h)(\nabla \cdot \boldsymbol{u}_h^{n+1,i}) \, \mathrm{d}\Omega = 0,
\end{aligned}$$

$$\begin{aligned}
&\int_\Omega q_h \nabla \cdot \boldsymbol{u}_h^{n+1,i} \, \mathrm{d}\Omega + \sum_{e=1}^{n_{\mathrm{el}}} \int_{\Omega^e} \tau_1^{n+1,i} \nabla q_h \cdot \Big[\frac{\rho}{\delta t} D_k \boldsymbol{u}_h^{n+1} \\
&\quad - \mu \, \Delta \boldsymbol{u}_h^{n+1,i} + \rho (\boldsymbol{u}_h^{n+1,i-1} \cdot \nabla)\boldsymbol{u}_h^{n+1,i} + \nabla p_h^{n+1,i} - \boldsymbol{f} \Big] \, \mathrm{d}\Omega = 0 \ ,
\end{aligned}$$

for $i = 1, 2, ...$until convergence, that is to say, until $\boldsymbol{u}_h^{n+1,i-1} \approx \boldsymbol{u}_h^{n+1,i}$and $p_h^{n+1,i} \approx p_h^{n+1,i-1}$ in the norm defined by the user.

The parameters $\tau_1$and τ_2 are chosen in order to obtain a stable numerical scheme with optimal convergence rates (see [24] and references therein for details). They are computed within each element domain Ω^e. We take them as:

$$\tau_1 = \left[\frac{4\mu}{(h^e)^2} + \frac{2\rho|\boldsymbol{u}^e|}{h^e} \right]^{-1}$$

$$\tau_2 = \frac{(h^e)^2}{\tau_1}$$

where h^e and $|\boldsymbol{u}^e|$ are a typical length and a velocity norm of element e, respectively. At least three option can be suggested for h^e: the maximum element length, the minimum one and the one in the direction of the flow. The strategy we are using in our code is to take the minimum element length for the diffusive term and one in the direction of the flow for the convective term. Thus to be more precise we should write [33],

$$\tau_1 = \left[\frac{4\mu}{(h^e_{min})^2} + \frac{2\rho|\boldsymbol{u}^e|}{h^e_{dir\_flow}} \right]^{-1}$$

$$\tau_2 = \frac{(h^e_{min})^2}{\tau_1}$$

1.2.5 Matrix version of the problem

In order to introduce some of the notation to be used in Chapter 4 we will present the matrix version of the problem using Picard linearization, BDF1 time discretization and split OSS stabilization.

The projections $P_h\left(\ \boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}\right)$ and $P_h\left(\frac{1}{\rho}\nabla p_h^{n+1}\right)$, will be treated iteratively using the same iterative loop as for the linearization of the convective term. We will call them

$$\boldsymbol{y}_h^{n+1}=P_h\left(\ \boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1}\right),$$
$$\boldsymbol{z}_h^{n+1}=P_h\left(\frac{1}{\rho}\left(\nabla p_h^{n+1}-\boldsymbol{f}^{n+1}\right)\right).$$

They are the solution of

$$\left(\boldsymbol{y}_h^{n+1},\boldsymbol{v}_h^*\right)=\left(\ \boldsymbol{u}_h^{n+1}\cdot\nabla\boldsymbol{u}_h^{n+1},\boldsymbol{v}_h^*\right)\qquad\forall\ \boldsymbol{v}_h^*\in\boldsymbol{V}_h^*,$$
$$\left(\boldsymbol{z}_h^{n+1},\boldsymbol{v}_h^*\right)=\left(\frac{1}{\rho}\left(\nabla p_h^{n+1}-\boldsymbol{f}^{n+1}\right),\boldsymbol{v}_h^*\right)\qquad\forall\ \boldsymbol{v}_h^*\in\boldsymbol{V}_h^*,$$

where $\boldsymbol{V}_h^*$ is the space $\boldsymbol{V}_h$ enlarged with the continuous vector functions associated to the boundary nodes.

The resulting algebraic system prior to linearization, supposing Laplacian form in order to simplify the presentation, is then

$$\mathsf{M}\frac{D_k}{\delta t}\mathsf{U}^{n+1}+\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}+\mathsf{G}\mathsf{P}^{n+1}+\mathsf{S}_u\left(\tau_1;\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}-\mathsf{S}_y\left(\tau_1;\mathsf{U}^{n+1}\right)\mathsf{Y}^{n+1}\qquad(1.10)$$
$$+\mathsf{S}_d\left(\tau_2\right)\mathsf{U}^{n+1}-\mathsf{S}_w\left(\tau_2\right)\mathsf{W}^{n+1}=\mathsf{F}^{n+1},$$
$$\mathsf{D}\mathsf{U}^{n+1}+\mathsf{S}_p\left(\tau_1\right)\mathsf{P}^{n+1}-\mathsf{S}_z\left(\tau_1\right)\mathsf{Z}^{n+1}=\mathsf{0},$$
$$\mathsf{M}_\pi\mathsf{Y}^{n+1}-\mathsf{C}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}=\mathsf{0},$$
$$\mathsf{M}_\pi\mathsf{Z}^{n+1}-\mathsf{G}_\pi\mathsf{P}^{n+1}=\mathsf{0},$$

where U, P, Y and Z are the arrays of the nodal unknowns for $\boldsymbol{u}$, p, $\boldsymbol{y}$ and $\boldsymbol{z}$, respectively. If we denote the node indexes with superscripts a,b, the space indexes with subscripts i, j and the standard shape functions of node a by N^a, the components of the arrays involved the previous equations are:

$$\mathsf{M}_{ij}^{ab}=\left(N^a,\rho N^b\right)\delta_{ij},$$

$$\mathsf{M}_{\pi}{}_{ij}^{ab} = \left(N^a, N^b\right) \delta_{ij},$$

$$\mathsf{K}\left(\mathsf{U}^{n+1}\right)_{ij}^{ab} = \left(N^a, \rho\, \boldsymbol{u}_h^{n+1} \cdot \nabla N^b\right) \delta_{ij} + \left(\nabla N^a, \mu \nabla N^b\right) \delta_{ij},$$

$$\mathsf{G}_i^{ab} = \left(N^a, \partial_i N^b\right),$$

$$\mathsf{G}_{\pi}{}_{i}^{ab} = \left(N^a, \partial_i N^b / \rho\right),$$

$$\mathsf{S}_u\left(\tau_1; \mathsf{U}^{n+1}\right)_{ij}^{ab} = \left(\tau_1 \boldsymbol{u}_h^{n+1} \cdot \nabla N^a, \rho\, \boldsymbol{u}_h^{n+1} \cdot \nabla N^b\right) \delta_{ij},$$

$$\mathsf{S}_y\left(\tau_1; \mathsf{U}^{n+1}\right)_{ij}^{ab} = \left(\tau_1 \boldsymbol{u}_h^{n+1} \cdot \nabla N^a, \rho N^b\right) \delta_{ij},$$

$$\mathsf{S}_d\left(\tau_2\right)^{ab} = \left(\tau_2 \nabla \cdot N^a, \nabla \cdot N^b\right),$$

$$\mathsf{S}_w\left(\tau_2\right)^{ab} = \left(\tau_2 \nabla \cdot N^a, N^b\right),$$

$$\mathsf{D}_j^{ab} = \left(N^a, \partial_j N^b\right),$$

$$\mathsf{S}_p\left(\tau_1\right)^{ab} = \left(\tau_1 \nabla N^a, \nabla N^b\right),$$

$$\mathsf{S}_z\left(\tau_1\right)_j^{ab} = \left(\tau_1 \partial_j N^a, \rho N^b\right),$$

$$\mathsf{C}\left(\mathsf{U}^{n+1}\right)_{ij}^{ab} = \left(N^a, \boldsymbol{u}_h^{n+1} \cdot \nabla N^b\right) \delta_{ij},$$

$$\mathsf{F}_i^a = \left(N^a, f_i\right),$$

where δ_{ij} is the Kronecker δ.

It is understood that all the arrays are matrices (except F, which is a vector) whose components are obtained by grouping together the left indexes in the previous expressions (a and possibly i) and the right indexes in the previous expressions (b and possibly j). Equation (1.10) needs to be modified to account for the Dirichlet boundary conditions (matrix G can be replaced by -D^T when this is done).

1.2.6 Material properties approximation

For the continuous problem definition, equation (1.6) is all that is needed to define the material properties for the two fluid Navier-Stokes equations. When the problem is discretized, the material properties ($\rho,\ \mu$) need to be somehow approximated in elements cut by the interface. The simplest approach is to take ς_k as either ς_1 or ς_2 depending

on the value of the level set function at each integration point (k), where ς stands for μ or ρ. The approximation depends on the integration rule used on elements cut by the interface. The typical approach is to use the same integration rule as on non cut elements. In Chapter 2 an enhanced integration rule for cut elements is presented. Another option can be found in [81].

Following the nomenclature used in the Level Set community [90] equation (1.6) is usually written as

$$\varsigma = \varsigma_1 + (\varsigma_2 - \varsigma_1)\, H(\psi),$$

where ς, as we have said, stands for μ or ρ , H is the Heaviside function,

$$H(\psi) = \begin{cases} 0 & \text{if } \psi \leq 0, \\ 1 & \text{if } \psi > 0, \end{cases}$$

and ψ is the level set function. We will therefore call this approach *Heaviside*.

The option most commonly used by Level Set practitioners is to smear-out the Heaviside function using

$$H_\epsilon(\psi) = \begin{cases} 0 & \text{if } \psi < -\epsilon, \\ \frac{1}{2} + \frac{\psi}{2\epsilon} + \frac{1}{2\pi}\sin\left(\frac{\pi\psi}{\epsilon}\right) & \text{if } -\epsilon \leq \psi \leq \epsilon, \\ 1 & \text{if } \psi > \epsilon, \end{cases}$$

where ϵ is a tunable parameter that determines the size of the bandwidth of numerical smearing. In [90] $\epsilon = 1.5\Delta x$ is suggested making the interface width equal to three grid cells, where Δx is the grid size for a finite difference discretization. It is interesting to note that in this case the material properties are also approximated in some elements not cut by the interface. We will call this approach *Smoothed Heaviside*.

In Section 1.1 we have said the *Level Set method* is also known as *pseudo–concentration technique* [110]. The only significant difference between the two methods is the way in which the material properties are approximated. A typical option found in works that use the *pseudo–concentration technique* is to calculate the material properties at integration points k belonging to cut elements according to

$$\varsigma_k = \xi_k\, \varsigma_1 + (1 - \xi_k)\, \varsigma_2.$$

The *pseudo–concentration* function, ξ, is for practical matters equivalent to the *level set* function and can be related to it according to

$$\xi = k_1\psi + k_2$$

where $k_1$and k_2 are two constants taken so that $0 \leq \xi \leq 1$. Usually $k_2 = 0.5$ is used. We will call this approach *Pseudo–concentration properties*.

1.2.7 Mixed boundary conditions on curved walls

In equation (1.4) we have defined mixed boundary conditions for the Navier Stokes equations as the conditions that prescribe a zero velocity in the normal direction to the boundary and a traction in the tangential direction. Two typical cases are slip boundary conditions where the tangential traction is zero and wall boundary conditions where the tangential traction depends on the velocity. For the numerical simulation of turbulent flows on complex geometries, such as the mould filling examples to solve in this thesis, the use of wall laws is mandatory since the simulation of the boundary layer is computationally too expensive.

In real problems one usually deals with domains with curved boundaries where mixed boundary conditions must be applied. When such domains are discretized using the finite element method, the normal to the different element faces belonging to the boundary that meet at a node do not coincide. In order to apply the zero normal velocity condition to the discretized problem, once the system that describes the problem without boundary conditions has been obtained in Cartesian coordinates, the degrees of freedom corresponding to nodes on the curved boundary must be rotated into a local system such that the normal component can be prescribed to zero. The problem with curved boundaries is that the normal defined at the node does not coincide with the normal to each of the element faces on the boundary that meet at the node. Therefore, the normal velocity to the faces will not be exactly zero and there will be some flow through the faces (the condition $\boldsymbol{n} \cdot \boldsymbol{u} = 0$ will not be exactly satisfied at the discrete level). The normal at the node is usually defined in such a way that the flow through the element faces on the

boundary associated with the velocity at the node is zero [48]. Such nodal normals are called consistent normals because they are the best solution to satisfy mass conservation.

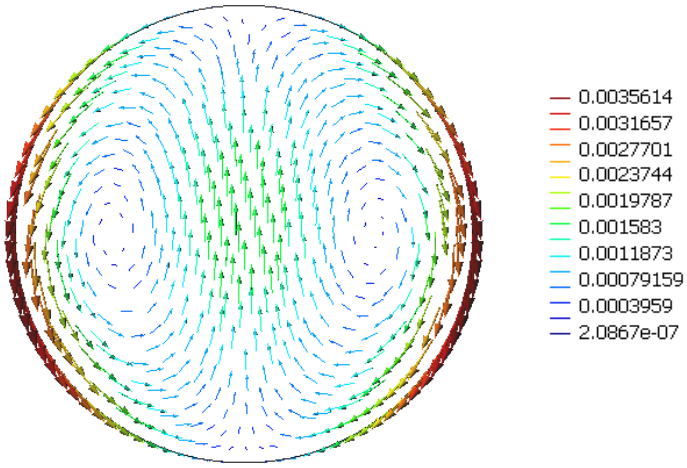


Figure 1.1: Spurious velocities obtained for a downward gravity force

Even when consistent normals are used, problems that have not been discussed until quite recently [9] can be observed. If one tries to simulate a fluid at rest under gravity forces in a domain with curved boundaries using mixed boundaries conditions spurious velocities will appear. Figure 1.1 shows the spurious velocities obtained in a circular domain of radius $r = 1.0$ with slip boundary conditions for $\rho = \mu = 1.0$ and $g = 10.0$ in the downward direction . For real flows these spurious velocities can be hidden by the real velocities and not be noticed. As the Froude number decreases they become more noticeable, specially on coarse meshes.

For constant density flows on fixed domains two very trivial solutions can be proposed. The first one is to replace the problem by an equivalent one where gravity forces disappear. The second solution is to integrate over the mixed boundary the traction corresponding to the hydrostatic pressure. In flows with variable density (such as two phase flows) the previous solutions cannot be applied. In [9] the proposed solution consists in applying a "do nothing" boundary condition [92] on mixed boundaries. Actually the normal component to the node is subtracted from the usual "do nothing" boundary conditions but this has no effect on the flow but only on the resulting reactions. Therefore unless one

is interested on the reactions on the boundary the usual "do nothing" boundary condition can be applied. In [9] only slip boundary conditions are discussed but the method could easily be extended to other mixed boundaries.

"Do nothing" boundary conditions consist in sending to the matrix side (instead of right hand side) the traction boundary conditions when tractions are not known. Despite at the continuous level this leads to an ill-posed problem, for the discrete finite element problem it typically yields very good results. It is commonly used at outflow boundaries where both normal and tangent components of the traction vector are unknown. For the mixed boundary cases we are analyzing the tangent components of the traction vector are known and therefore we belive that the "do nothing" boundary condition should be applied only in the normal direction and not in all the directions as proposed in [9]. That is to say, instead of integrating (on the matrix side)

$$-\int_{\Gamma_{\mathrm{mu}}} \boldsymbol{v}\cdot\left[-p\boldsymbol{n}+\mu\left(\boldsymbol{n}\boldsymbol{\cdot}\left(\boldsymbol{\nabla u}+(\boldsymbol{\nabla u})^{t}\right)\right)\right]\ \mathrm{d}\Gamma=-\int_{\Gamma_{\mathrm{mu}}}\boldsymbol{v}\cdot[\boldsymbol{n}\cdot\boldsymbol{\sigma}]\ \mathrm{d}\Gamma$$

as proposed in [9], one should only integrate

$$-\int_{\Gamma_{\mathrm{mu}}}\boldsymbol{v}\cdot[(\boldsymbol{n}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{n})\,\boldsymbol{n}]\ \mathrm{d}\Gamma$$

For a flow at rest we have obtained the exact solution with both formulations. For a problem with non zero velocities, the first condition we must expect from the proposed formulations is that in the case with planar boundaries the solution obtained without any modification should remain unaltered when the modification for curved boundaries is applied. This is what happens when the open boundary condition in the normal direction we propose is applied. When the modification proposed in [9] is used the boundary conditions in the tangential direction are altered and so is the solution.

In order to show what happens in a numerical example, the test case we propose is Stokes flow in a channel. Taking into account symmetry only half of the channel is simulated. On the symmetry face a slip boundary condition is applied. Despite the test case does not have curved boundaries, the proposed correction should also work on planar boundaries. Moreover, the absence of curved boundaries allows us to solve the problem

with no special correction. In Figure 1.2 we compare the solutions obtained with both corrections against the solution obtained without any correction. The results obtained without any boundary correction are exactly the same as those obtained with the "do nothing" boundary condition applied only in the normal direction. On the other hand the use of the "do nothing" boundary condition in all directions [9] leads to an incorrect solution.

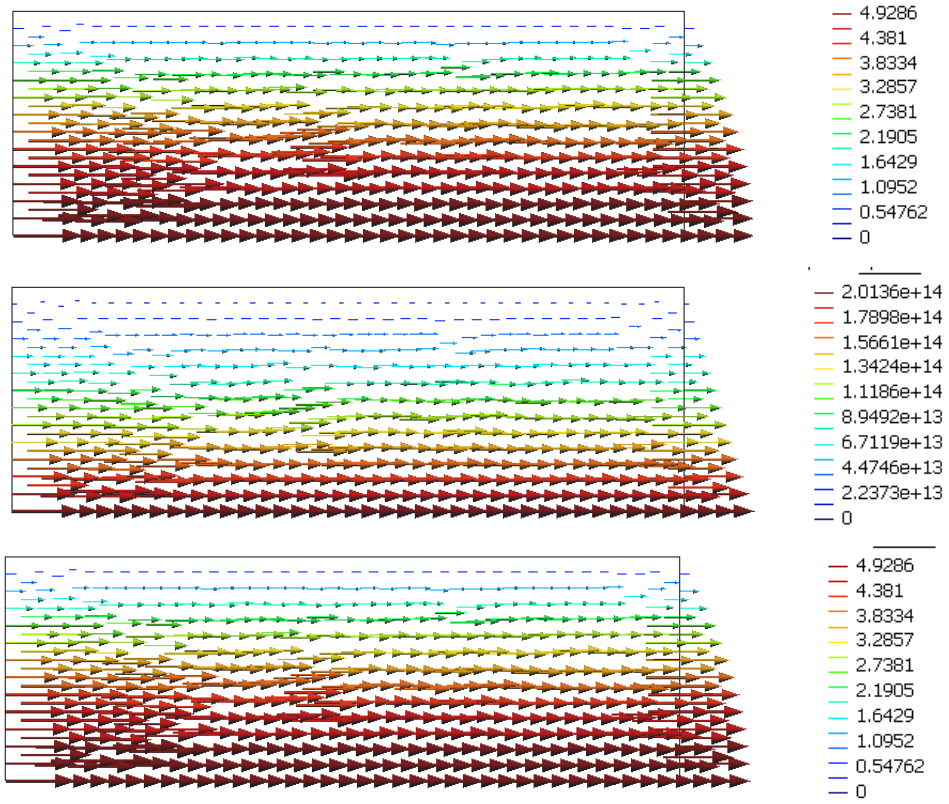


Figure 1.2: Velocities in a channel: (top) no modification, (middle) modification proposed in [9], (bottom) our modification

The test case that has been presented is an extreme case to show the improvement introduced by our modification to the method proposed in [9]. Since the implementation and computational cost of both methods is nearly the same we believe the "do nothing" BC only in the normal direction should be used. In the mould filling examples shown in Chapter 5 this correction has been a key element for obtaining correct solutions.

1.3 The Level Set equation

1.3.1 Interface Capturing Techniques

As we have already mentioned, fixed mesh methods generally share two basic steps. In the previous Section we have presented the two fluid Navier–Stokes equations that allow us to deal with the first step. In this Section we present an equation for an interface function that allows to determine the position of the interface, and thus the properties to be assigned in the previous step.

Several methods can be used to determine the position of the interface. We will use the so called *level set* method (see [18, 106] and [90] for an overview) but other methods could also have been used. Among those more closely related to the *level set* method we would like to mention the *pseudo–concentration technique* [110] and *volume of fluid* (VOF) technique [57, 70, 103]. Whether they are different methods or different names for the same method is a question that can raise some discussion.

In all three methods, a transport equation,

$$\partial_t \phi + (\boldsymbol{u} \cdot \nabla)\phi = 0$$

for a function that coincides with the name of the method is solved. In the case of the level set the function we will use the letter ψ for ϕ. The basic idea of the level set method is to embed the propagating interface $\Gamma_{\text{int}}(t)$ as the zero level set of a higher dimensional function ψ, defined as $\psi(\boldsymbol{x}, t=0) = \pm d$, where d is the distance from $\boldsymbol{x}$ to $\Gamma_{\text{int}}(t=0)$, chosen to be positive in one fluid and negative in the other. Setting the initial zero level set so that it coincides with the initial interface $\Gamma_{\text{int}}(t=0) = \{\boldsymbol{x} \mid \psi(\boldsymbol{x}, t=0) = 0\}$, the previous transport equation contains the embedded motion for $\Gamma_{\text{int}}(t)$ as the level set $\psi = 0$. It is interesting to remark that ψ is a continuous function.

In the *pseudo–concentration* method, the scalar function ϕ can be considered a fictitious fluid property that is advected by the flow and indicates the presence of fluid one or fluid two at a certain point $\boldsymbol{x}$. In this sense it is equivalent to the level set method. Typically $\phi = 1$ is assigned to the region occupied by fluid one and $\phi = 0$ to the region

occupied by fluid two. In this sense the *pseudo–concentration* method is similar to the VOF method we will describe next. The position of the front is defined by the isovalue contour $\phi(\boldsymbol{x}) = \phi_c$, where $\phi_c \in [0,1]$ is a critical value defined a priori (typically $\phi_c = 0.5$). This critical value used is irrelevant before the problem is discretized, but is needed when the finite element discretization is introduced. The transport of a function that varies abruptly from 0 to 1 introduces problems when solved numerically, therefore it is usually smoothed redefining the *pseudo–concentration* for each node of the finite element mesh according to the following expression [110]:

$$\phi = \phi_c + sgn(\phi_0 - \phi_c)\ k\ d$$

where ϕ_0 is the non-smoothed value, k is a constant, d is the distance from the point under consideration to the front and $sgn(\cdot)$ is the signum of the value enclosed in brackets. Then the only differences with the level set function are that the pseudo-concentration slope can have any value and that the critical value used is 0.5 instead of 0. Those two differences have no practical importance and therefore the two methods can be considered identical, at least in the way in which they capture the interface (some differences in the way they approximate the material properties close to the interface have been discussed in the previous Section). It is interesting to remark that the *pseudo–concentration* method appeared before the *level set* method [91]. The latter method has obtained more widespread use thanks to its clear presentation and mathematical formalism. In any case we consider that it is the same method with two different names.

In the VOF method the scalar ϕ represents the fractional volume of a certain fluid in the corresponding computational cell. Typically $\phi = 1$ is assigned to cells occupied by fluid one and $\phi = 0$ to cells occupied by fluid two. Cells cut by the front have some $\phi \in [0,1]$ depending on the percentage of each fluid present. The VOF is typically associated to discretizations with constant interpolations within each cell, such as finite volumes. The following analogy can be proposed: if Finite Volumes are seen an P0 Finite Elements, then the VOF method can be seen as a P0 discretization of the *pseudo–concentration* or *level set* method. One particularity of the VOF method is that the position of the interface

is not obtained directly from the value of ϕ, but reconstructed from it. Several interface reconstruction techniques can be found in the literature (see [70, 103, 116] for reviews). Another particularity is that in most versions of the VOF method the transport equation is not discretized and solved algebraically but rather solved geometrically.

Since we are using a finite element discretization the *level set* method is the natural choice, but different discretizations for the Navier-Stokes and interface capturing equations could also be used.

1.3.2 Implementation of the level set method

As we have already mentioned, the basic idea of the level set method is to define a smooth scalar function, say $\psi(\boldsymbol{x}, t)$, over the computational domain Ω that determines the extent of subdomains Ω_1 and Ω_2 and allows to represent the interface implicitly. For instance, we may assign positive values to the points belonging to Ω_1 and negative values to the points belonging to Ω_2 . The position of the fluid front is then defined by the iso-value contour $\psi(\boldsymbol{x}, t) = 0$. The evolution of the front $\psi = 0$ in any control volume $V_t \subset \Omega$ which is moving with a divergence free velocity field $\boldsymbol{u}$ leads to:

$$\partial_t \psi + (\boldsymbol{u} \cdot \nabla)\psi = 0 \tag{1.11}$$

This equation is hyperbolic and therefore boundary conditions for ψ have to be specified at the inflow boundary, defined as:

$$\Gamma_{\text{inf}} := \{x \in \partial\Omega \quad | \quad \boldsymbol{u} \cdot \boldsymbol{n} = 0\}$$

Function ψ is the solution of the hyperbolic equation (1.11) with the boundary conditions:

$$\psi = \overline{\psi} \qquad \text{on } \Gamma_{\text{inf}} \times (t_0, t_f),$$

$$\psi(\boldsymbol{x}, 0) = \psi_0(\boldsymbol{x})$$

The initial condition ψ_0 is chosen in order to define the initial position of the fluid front to be analyzed. The boundary condition $\overline{\psi}$ determines which fluid enters through a certain point of the inflow boundary.

In order to obtain the weak or variational formulation of the Level Set equation (1.11) we introduce the spaces

$$X_0 \equiv \left\{ x \in L^2(\Omega) \mid x = 0 \text{ on } \Gamma_{\text{inf}} \right\},$$

$$X \equiv \left\{ x \in L^2(\Omega) \mid x = \overline{\psi} \text{ on } \Gamma_{\text{inf}} \right\},$$

$$X_t \equiv L^2(t_0, t_f; X),$$

The weak form is then obtained by multiplying the transport equation for the front (1.11) by an arbitrary element of X_0, x.

The weak form of problem 1.11 with the boundary conditions we have just defined is: Find $\psi \in X_t$ such that

$$\int_\Omega \partial_t \psi \; x \; \mathrm{d}\Omega + \int_\Omega \left[\boldsymbol{u} \cdot \nabla \psi \right] \; x \; \mathrm{d}\Omega = 0$$

for all $x \in X_0$.

The spatial discretization is the same one used for the Navier-Stokes equation, that is P1 elements. The temporal evolution is treated via the backward differencing (BDF) time integration scheme. Due to the pure convective type of equation (1.11), we use the SUPG [14] technique to stabilize it. Since equation (1.11) does not have a diffusive term, the use the SUPG technique is equivalent to the use of the ASGS technique presented in the previous chapter.

The discrete problem, both in space and time, stabilized using the SUPG or ASGS technique then reads: Given a velocity $\boldsymbol{u}_h^{n+1}$ at time t^{n+1} and a ψ_h^n at time t^n (and also at previous times as required by D_k), find $\psi_h^{n+1} \in X_h$ (a discrete linear subspace of X) by solving the discrete variational problem:

$$\int_\Omega \frac{1}{\delta t} D_k \psi_h^{n+1} \; x_h \; \mathrm{d}\Omega + \int_\Omega \left[\boldsymbol{u}_h^{n+1} \cdot \nabla \psi_h^{n+1} \right] \; x_h \; \mathrm{d}\Omega$$
$$+ \sum_{e=1}^{n_{\text{el}}} \int_{\Omega^e} \tau^{n+1} \left[\boldsymbol{u}_h^{n+1} \cdot \nabla x_h \right] \left[\frac{1}{\delta t} D_k \psi_h^{n+1} + \boldsymbol{u}_h^{n+1} \cdot \nabla \psi_h^{n+1} \right] \mathrm{d}\Omega = 0$$

As in the previous chapter, the parameter τ is chosen in order to obtain a stable numerical scheme with optimal convergence rates. It is computed within each element domain Ω^e

as:

$$\tau = \frac{h^e}{2|\boldsymbol{u}^e|},$$

where h^e is the element length in the direction of the flow and $|\boldsymbol{u}^e|$ the velocity norm of element e.

1.3.3 Reinitialization

For the numerical solution of the level set equation it is preferable to have a function without large gradients. Since the only requirement such a function must meet is $\psi = 0$ at the interface, a signed distance function ($\mid \nabla\psi \mid = 1$) is used. Under the evolution of the level set equation, ψ will not remain a signed distance function and thus needs to be reinitialized. Several approaches can be found in the literature [18, 83, 90, 104, 106]. The one we will use consists in redefining ψ for each node of the finite element mesh according to the following expression:

$$\psi = \mathrm{sgn}(\psi^0)d$$

where ψ^0 stands for the calculated value of ψ, d is the distance from the node under consideration to the front, and $\mathrm{sgn}(\cdot)$ is the signum of the value enclosed in the parenthesis.

In [21] three ways of calculating d are discussed. There the one we will use here is called 'interpolation of a straight line' and is described briefly for the case of linear elements. The free surface is approximated by triangular planes p (lines in 2D). Then the perpendicular distance d_ip of each grid point i to each plane p can be computed. The minimum distance from each nodal point to the planes is the required distance between the point and the front ($d_i = \min_{\mathrm{p}}\{d_ip\}$). For the bigger examples we have reinitialized the signed distance function only on five layer of nodes to each side of the interface to reduce the computational cost. Another option can be found in [32].

The approach we use is typically not favored in the level set community because it is considered slow [90]. A great deal of effort has been put into reinitialization techniques and a complete review of most recent advances can be found in [90]. Most of them are

based in solving an equation of the form

$$\partial_t \psi + |\nabla \psi| = 1$$

up to steady state, where t is not the real time but some pseudo time. When the steady state is reached the transient term disappears and then the signed distance function ($\mid \nabla \psi \mid = 1$) is recovered. Although most of the options presented seem very promising, for the moment we do not feel it worth for us to put an effort into the subject because despite our reinitialization technique may be slow, it is, according to what we have observed, not the key item that makes our method slow. On the other hand, as we have already mentioned a lot work has been done on the subject and our initial effort should, in any case, concentrate in testing some of the available options [32].

Another technique that is somehow related to reinitialization is the construction of extension velocities [90, 104]. The range of use of the level set technique is much wider than flows with interfaces. There are lots of application where the velocity is only defined on the interface and not in the rest of the domain. In theory, the only velocities needed to transport the interface are those on the interface, but since in the level set method the interface is transported embedded in a higher dimensional function the velocities in the rest of the domain have to be found from the velocities on the interface. Such velocities are called extension velocities. In the case of flow simulations the use of extension velocities is not mandatory but it has been suggested that using only the real velocities on the interface and extension velocities in the rest of the domain can help to maintain the level set a signed distance function, thus reducing the need for reinitialization [104].

1.3.4 Coupling between the flow equations and the Level Set

In the previous Section we have presented the two fluid Navier-Stokes equations and in this Section we have dealt with the Level Set equation. Obviously both equations are coupled. In the flow equations the fluid properties (ρ and μ) depend on the level set function, ψ_h^{n+1}. On the other hand, the level set function is transported using the velocity $\boldsymbol{u}_h^{n+1}$. The approach we will use to uncouple them is to use the $\boldsymbol{u}_h^n$ (or $2\boldsymbol{u}_h^n - \boldsymbol{u}_h^{n-1}$ when

a second order backward difference is used) instead of $\boldsymbol{u}_h^{n+1}$ when calculating ψ_h^{n+1}. This introduces a restriction on the time step size so that the front does not advance more than one layer of elements at a time. In any case advancing more elements at a time might make us lose the physics of the problem and therefore we believe this time step restriction is logical.

Another option could be to use a block iterative procedure to couple both equations. Then the velocity to be used in transport of the level set would be $\boldsymbol{u}_h^{n+1,i}$ where the superscript i refers to the iteration number. It could be the same iterative loop used for the non-linearity introduced by the convective term of the Navier-Stokes equations or some other purposely introduced iterative loop. In [21] such strategy has been tested but the uncoupled option has been preferred.

Chapter 2

An enriched pressure two-phase flow model

As we have mentioned in Chapter 1, numerical methods for flows with interfaces can be classified into fixed grid and moving grid techniques. For the same mesh size the latter lead to a more accurate representation of the interface at a higher computational cost. In this paper we will use a fixed grid method and introduce modifications to the basic formulation to enhance the representation of the interface.

The initial motivation for this work came from the impossibility to model a terribly simple flow; in fact, not even a flow, but two different density fluids at rest (the lighter one on top) inside a closed cavity. The hydrostatic pressure gradient is discontinuous at the interface. This cannot be correctly represented by the usual finite element shape functions when the front crosses an element. Since the velocity and pressure are coupled, the error in the representation of the pressure gives rise to spurious velocities that, depending of the properties used, can completely distort the interface that should otherwise remain horizontal.

The idea of enriching the representation of an unknown at a material discontinuity is not new and several approaches can be found in the literature [19, 45, 81]. In the next Section we will outline some of the differences between the previous methods and ours.

As we have already mentioned for the evolution of the fluid interface, we use the *level set* method [18, 106]. The contribution we intent to introduce in this chapter does not depend on the approach used to capture the interface and should also be valid using any of the other cited techniques.

The numerical formulation presented here to solve the incompressible Navier–Stokes equations uses a time discretization based on the standard trapezoidal rule and a stabilized finite element method referred to as Algebraic Sub-Grid Scales (ASGS) [24]. P1-P1 elements will be used. In the elements cut by the interface the P1 pressure shape functions are supplemented with an additional shape function that is zero at all the element nodes, continuous within the element and has a constant gradient on each side of the interface. This shape function is local to each element and the corresponding degree of freedom can therefore be condensed prior to assembly, making the implementation quite simple on any existing finite element code. The details will be discussed in the next Section.

In Chapter 1 we have described the mathematical model used to solve the Navier–Stokes equations when no enrichment functions are used and the Level Set Method. In the next Section the enrichment functions used and some implementation details are discussed. Finally we present three simple numerical examples where the improvements obtained with the proposed formulation show up clearly.

The work we will describe in this chapter has been presented in [37]. Further numerical examples can be found in [36].

2.1 Discontinuous Gradient Pressure Shape Functions

In fixed grid finite element methods the whole domain Ω is subdivided into elements Ω^e. Within each element the unknowns are interpolated as

$$\phi_h|_{\Omega^e} = \sum_{I=1}^{\text{NNODE}} N_e^I \Phi_e^I,$$

where NNODE is the number of element nodes.

In typical finite element methods, ∇N_e^I are continuous within each element and therefore $\nabla \phi_h|_{\Omega^e}$ is continuous. When the interface crosses an element the discontinuity in the material properties leads to discontinuities in the gradients of the unknowns that the interpolation used cannot capture. For example, as mentioned previously, for two different density fluids at rest the interpolation errors in the pressure give rise to spurious velocities that can render the solution meaningless. Also, viscosity discontinuities can lead to discontinuous velocity gradients.

Enrichment methods add degrees of freedom at elements cut by the interface in order to reduce interpolation errors. In our particular case we add only one pressure degree of freedom per cut element. Therefore the pressure in elements cut by the interface is interpolated as

$$p_h|_{\Omega^e} = \sum_{I=1}^{\text{NNODE}} N_e^I P_e^I + N_e^{\text{ENR}} P_e^{\text{ENR}}. \tag{2.1}$$

The shape function N_e^{ENR} we introduce has a constant gradient on each side of the interface, its value is zero at the element nodes and is C^0 continuous in Ω^e. The added degree of freedom is local to the element and can therefore be condensed after the element matrix has been computed and before assembly. The resulting pressure finite element space is made of functions that are discontinuous across interelement boundaries, and thus it is a subspace of $L^2(\Omega)$, but not of $H^1(\Omega)$, as would be the case using $P1 - P1$ elements. However, our method is still conforming. If we had tried to use the previous enrichment functions for the velocity we would have obtained a non conforming method.

Minev and co workers [81] also use enrichment functions for two–fluid flows. They use discontinuous gradient velocity shape functions and discontinuous pressure shape functions because they include surface tension. No discontinuous gradient pressure shape functions are included. The velocity enrichment functions are not only local to each element, as in our case, but also continuous across elements. This is possible because they are the product of a bubble function times some other function. Tempted by the fact of being able to condense the enriched degree of freedom while using H^1 functions, we initially tried to use the enrichment they use for the velocity for the pressure, but

obtained very unsatisfactory results in the hydrostatic two–fluid case. This led us to look for enrichment functions that could model constant gradients on each side of the interface.

Also for two–fluid problems, Chessa and Belytschko [19] use an enrichment method called XFEM, initially developed by the second author for modeling cracks. As in the case of Minev et al., they use discontinuous gradient velocity shape functions and not discontinuous gradient pressure shape functions. The enrichment they use is continuous and cannot be condensed prior to matrix assembly. It is computationally much more expensive because the mesh graph needs to be updated as the interface moves from one element to another.

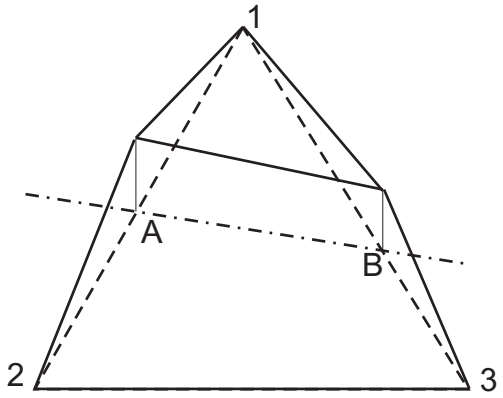


Figure 2.1: 2D Enrichment function for a cut element

In Fig. 2.1 we show a sketch of the enrichment function we use for an element cut by the interface in the 2D case. The element has nodes named 1, 2 and 3 and the interface cuts the element edges at points A and B. A way to build such function is as follows. Suppose that node 1 belongs to Ω_1 and nodes 2 and 3 belong to Ω_2. Let $\Omega_1^e = \Omega_1 \cap \Omega^e$ and $\Omega_2^e = \Omega_2 \cap \Omega^e$. In Ω_2^e we want N^{ENR} to have constant gradient and to have a zero value at $\boldsymbol{x}_2$ and $\boldsymbol{x}_3$. We can therefore define

$$N^{\text{ENR}}\big|_{\Omega_2^e} = k^1 \;\; N^1\big|_{\Omega_2^e},$$

where k^1 is a constant to be defined. By definition we want $N^{\text{ENR}}(\boldsymbol{x}_A) = 1$. As we are

using linear elements to interpolate the level set function we have that

$$N^1(\boldsymbol{x}_A) = \frac{\Psi^2}{\Psi^2 - \Psi^1},$$

where Ψ^i is the value of ψ at node i, and therefore

$$k^1 = \frac{\Psi^2 - \Psi^1}{\Psi^2}.$$

Now we have k^1 we can find

$$N^{\mathrm{ENR}}(\boldsymbol{x}_B)|_{\Omega_2^e} = k^1 \;\; N^1(\boldsymbol{x}_B)\big|_{\Omega_2^e} = k^1 \;\frac{\Psi^3}{\Psi^3 - \Psi^1}.$$

We can proceed to find $N^{\mathrm{ENR}}|_{\Omega_1^e}$. We want it to have a constant gradient in Ω_1^e and to be zero at $\boldsymbol{x}_1$. Then

$$N^{\mathrm{ENR}}\big|_{\Omega_1^e} = k^2 \;\; N^2\big|_{\Omega_1^e} + k^3 \;\; N^3\big|_{\Omega_1^e}.$$

Using once more that $N^{\mathrm{ENR}}(\boldsymbol{x}_A) = 1$ and the fact that $N^3(\boldsymbol{x}_A) = 0$ we get

$$k^2 = \frac{1}{N^2(\boldsymbol{x}_A)} = \frac{\Psi^1 - \Psi^2}{\Psi^1}.$$

Since we want the enrichment function to be continuous in Ω^e we need

$$N^{\mathrm{ENR}}(\boldsymbol{x}_B)|_{\Omega_2^e} = N^{\mathrm{ENR}}(\boldsymbol{x}_B)|_{\Omega_1^e},$$

then, as $N^2(\boldsymbol{x}_B) = 0$,

$$k^3 = N^{\mathrm{ENR}}(\boldsymbol{x}_B)|_{\Omega_2^e} \frac{1}{N^3(\boldsymbol{x}_B)} = k^1 \;\frac{\Psi^3}{\Psi^3 - \Psi^1}\frac{\Psi^1 - \Psi^3}{\Psi^1},$$

$$k^3 = -k^1 \;\frac{\Psi^3}{\Psi^1}.$$

We have obtained an enrichment function that is proportional to N^1 on Ω_2^e and a linear combination of N^2 and N^3 on Ω_1^e, where the values of k^1, k^2, k^3 only depend on the values of the level set function at the element nodes. It is very easy to obtain the enrichment function and its Cartesian derivatives from the usual shape function. It seems worthwhile

to remark that $N^{\text{ENR}}|_{\Omega^e}$ does not belong to the space formed by $N^1|_{\Omega^e}$, $N^2|_{\Omega^e}$, $N^3|_{\Omega^e}$. The same ideas have been used to obtain N^{ENR} for 3D elements.

In order to capture the discontinuities and take advantage of the enrichment functions used, the integration rules need to be modified in elements cut by the front. The method we use is to divide each tetrahedral (triangular in 2D) element into up to six tetrahedral (three triangular in 2D) sub elements. For each sub element the same integration rule as for the non–cut elements is used (see Figure 2.1).

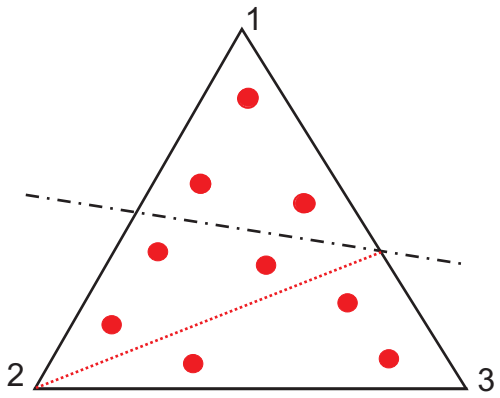


Figure 2.2: Enhanced integration rule for a 2D cut element

When using enrichment functions for the pressure, the material properties μ, ρ are taken as μ_1, ρ_1 or μ_2, ρ_2 depending on which part of the domain (Ω_1 or Ω_2) the integration point is found, that is, the material properties are approximated using the approach called *Heaviside* in Chapter 1.

Since the pressure space is enriched, a remark is needed concerning pressure stability. If we had used a velocity-pressure interpolation satisfying the inf-sup condition, the enrichment of the pressure could have led to an unstable velocity-pressure pair. However, we are using a stabilized finite element formulation. Even though we have no stability analysis for the enriched pressure space, we have not encountered any type of stability misbehavior.

A final remark is required concerning the extension of the proposed enrichment to higher order elements. Since the intention is to add a pressure field able to deal with

discontinuous pressure gradients, but constant in each fluid phase, exactly the same methodology as described for P_1 elements can be applied to higher order elements. The construction of the enriched pressures can be based only in the linear part of the interpolation basis functions of these higher order elements. This is particularly simple when they are implemented using a hierarchical basis. The case of quadrilateral elements (or hexahedra in 3D) can be dealt with by splitting the quadrilateral into triangles (or tetrahedra).

2.2 Numerical Examples

In this section we present three numerical examples where the improvements obtained with the proposed formulation show up clearly. The first two examples are related to the original two–fluid hydrostatic problem but modified so that they have nonzero velocities. The results obtained with the enriched formulation are compared with those obtained with a typical finite element formulation with no enrichment nor improved integration. Since we are trying to prove that the benefits come from the pressure enrichment and not from the improved integration, an intermediate case where no enrichment is used and we only modify the integration is also presented. Regarding the approximation of the material properties in elements cut by the interface, we have mentioned three possibilities in Chapter 1, *Heaviside, Smoothed Heaviside* and *pseudo-concentration* properties. When we use improved integration, with or without enriched pressures, the natural choice is to use *Heaviside* properties. When no improved integration nor pressure enrichment is used any of the three options can be used. In the examples we present in this chapter we have used *Heaviside* or *pseudo-concentration* properties. *Smoothed Heaviside* properties have been tested but the results were similar to those obtained with the other properties approximations and will not be presented.

Mesh	Npoin	Nelem	Elem length
coarse	128	214	1.0
medium	472	862	0.5
very fine	11615	22828	0.1

Table 2.1: Meshes used for the two fluid cavity flow

2.2.1 Two–fluid cavity

The first example, called the two–fluid cavity, is a square domain filled with equal amounts of two different density fluids and a fixed horizontal velocity (0.1 m/s) on the bottom wall. The walls are supposed frictionless. *Heaviside* properties are used to approximate the material properties in elements cut by the interface. It is a very simple example but it can be representative of the numerical problems that can appear in much more complex problems such as the two–phase flow in a stirred reactor.

Three 2D unstructured triangular meshes were used (see Table 2.1) . The square domain has a side length $L = 10\ m$. The material properties used (SI units) are $\rho_1 = 1000$, $\mu_1 = 10$ for the fluid on the bottom, and $\rho_2 = 900$, $\mu_2 = 9$ for the one on top. The viscosity of the bottom fluid is 1000 times the viscosity of water so as to obtain a relatively low Reynolds number ($Re = 100$) in order to avoid unnecessary complications. The simulations where run for 100 seconds with a 0.5 second time step size. In all the examples presented in this paper the acceleration of gravity is $g = 10$.

In Fig. 2.3 we show the shape of the interface for the three meshes in the three different conditions mentioned previously: with no enrichment nor improved integration, using only modified integration but no pressure enrichment and finally with both pressure enrichment and improved integration. Using the finest mesh the three methods give nearly the same result. Only in the case with no enrichment nor improved integration, slight oscillations can be observed. Despite we have not got physical measurements, the solutions with this mesh can be taken as a reference against which we can compare the results obtained with

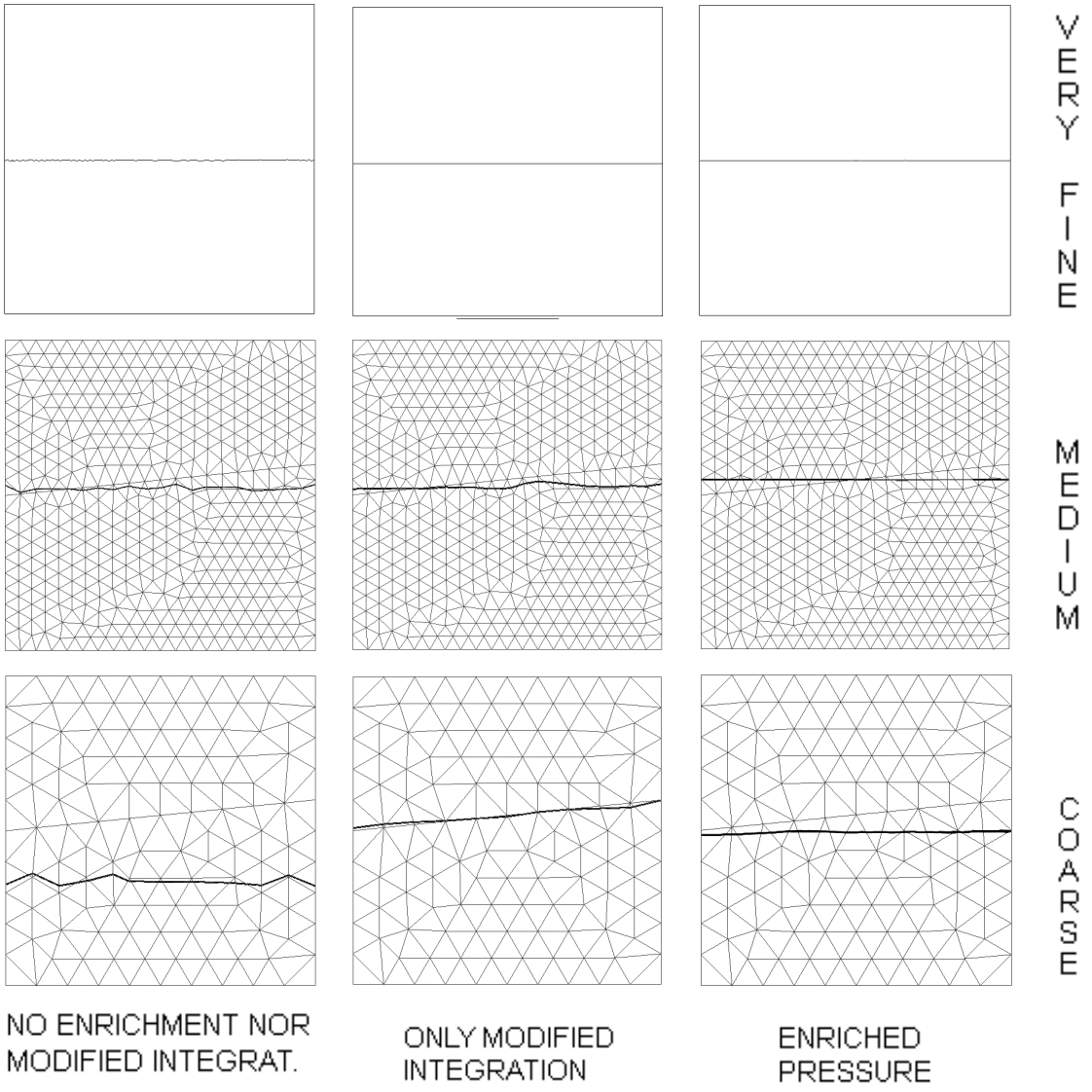


Figure 2.3: Shape of the interface for the different meshes and numerical conditions at $t = 100s$

the other two meshes. Using the medium mesh, only the simulation with both pressure enrichment and improved integration attains results nearly as good as those obtained with the very fine mesh. The other two cases show a distorted interface shape. Finally, using the coarse mesh, the shape of the interface using pressure enrichment and improved integration shows slight errors but is much better than the other two cases, where it can be clearly observed that mass is not conserved.

The flow pattern is compared in Fig. 2.4. Using the finest mesh there is not much difference between the three methods. Two recirculations can be found, one in the bottom fluid and one in the top one. With the medium mesh, the results obtained with pressure enrichment remain very similar to the ones obtained with the previous mesh. In the other two cases the flow pattern is strongly modified by the errors that originate close to the interface. With the coarse mesh the results deteriorate in all three cases as expected, but it can be observed that in the case with pressure enrichment the solution is still better than that obtained with a medium mesh in the other two cases. Even though the global flow pattern obtained with the very fine mesh is nearly the same in all three cases, a zoom at the velocity vectors close to the interface (see Fig. 2.5) reveals that even in this case, the pressure enrichment produces a better solution. The spurious oscillatory behavior obtained close to the interface in the cases without pressure enrichment has been observed not only in space but also in time. The effect of reducing the time step size has been analyzed, but no significant improvements have been obtained compared to those resulting from the pressure enrichment.

Finally, it has been observed that the convergence in the L^2 velocity norm within each time step is much better using the enriched formulation than without it, for all three meshes. Using a 0.0001 relative convergence tolerance for the velocity, the two formulations without enrichment converge in twice or more iterations than the enriched one. The enriched case takes 2 iterations to converge with the fine mesh, 3 with the medium mesh, and 4 with the coarse one.

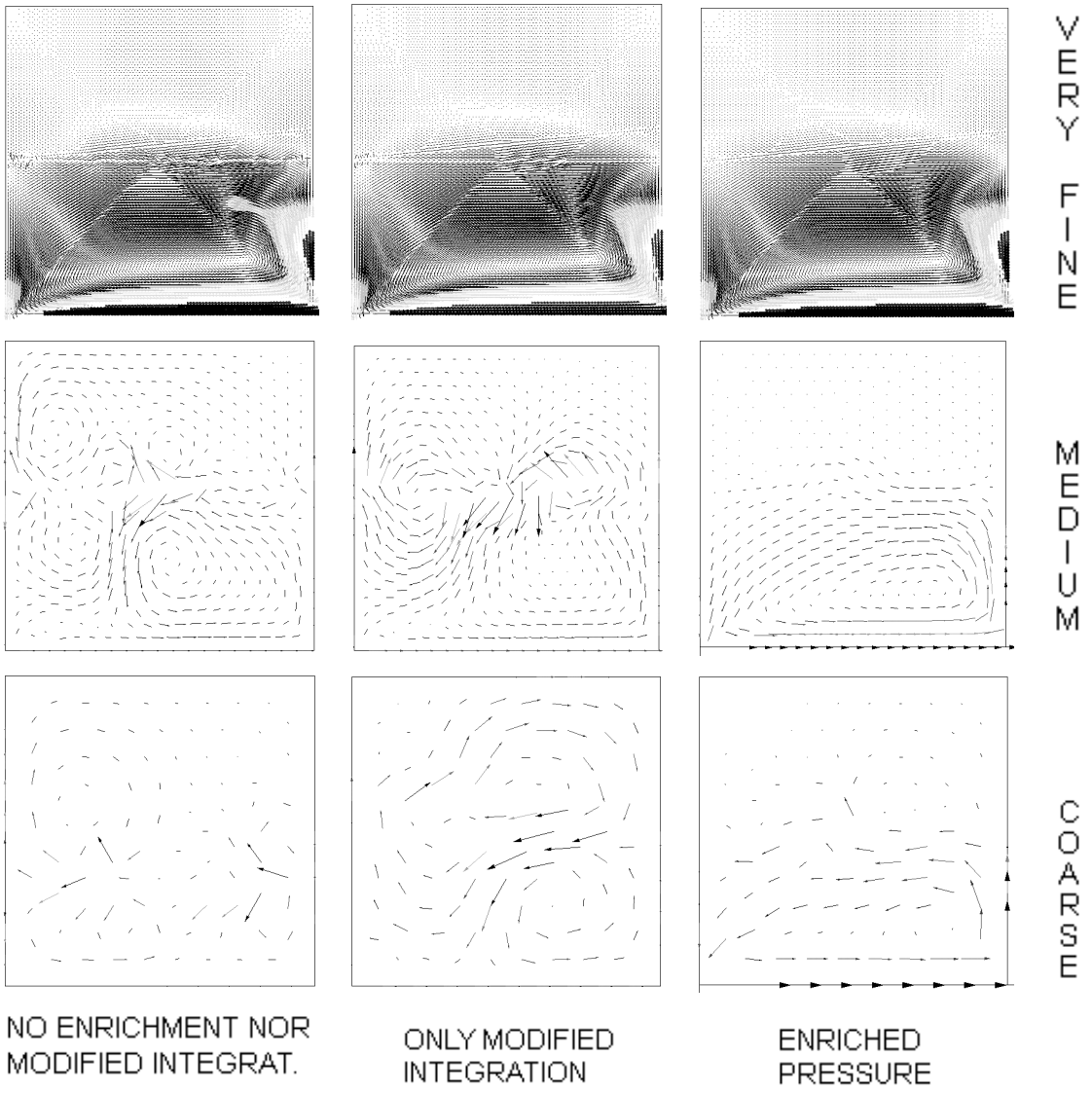


Figure 2.4: Flow pattern for the different meshes and numerical conditions at $t = 100s$

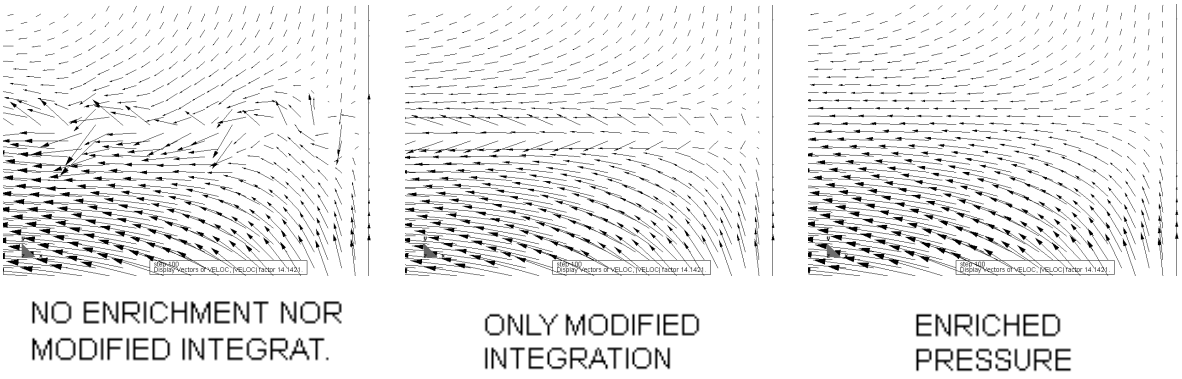


Figure 2.5: Detailed flow pattern for the finest meshes and different numerical conditions close to the interface

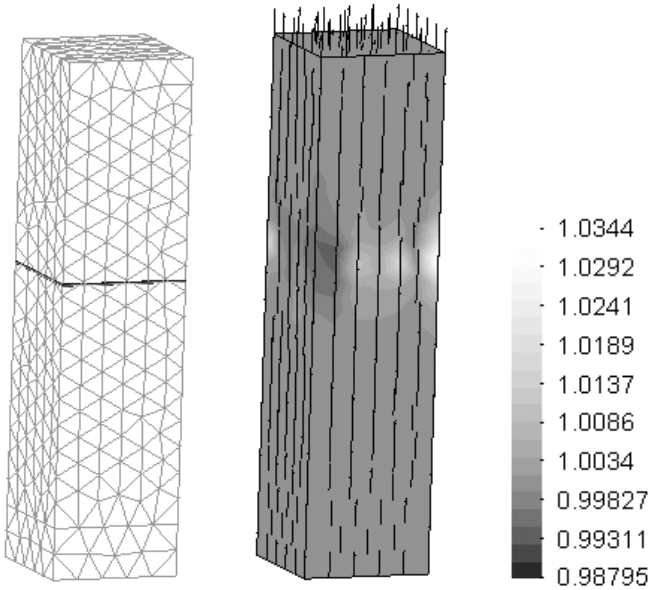


Figure 2.6: Interface shape and vertical velocity band plot together with velocity vectors using pressure enrichment

2.2.2 3D vertical channel

The second example is a 20 m high vertical channel with a square cross section (side length $L = 5\ m$). The channel is fed from the bottom with a heavier fluid at a constant (both in space and time) 1 m/s velocity and the upper face is left free so that the lighter fluid can escape. No friction is assumed on the walls. The initial interface is flat and at 2.5 m from the entrance. The solution for this problem is very simple. The velocity in the whole domain, included the interface, should be equal to the inlet velocity and the interface should remain flat.

A 3D unstructured tetrahedral mesh with 1106 nodes and 4921 elements is used. The material properties used (SI units) are $\rho_1 = 1000$, $\mu_1 = 100$ for the fluid on the bottom, and $\rho_2 = 10$, $\mu_2 = 1$ for the one on top. The time step size is 0.1 s. The Reynolds number based on the length of the square section is $Re = 50$. In the case with no improved integration nor pressure enrichment both *Heaviside* or *pseudo-concentration* properties have been tested.

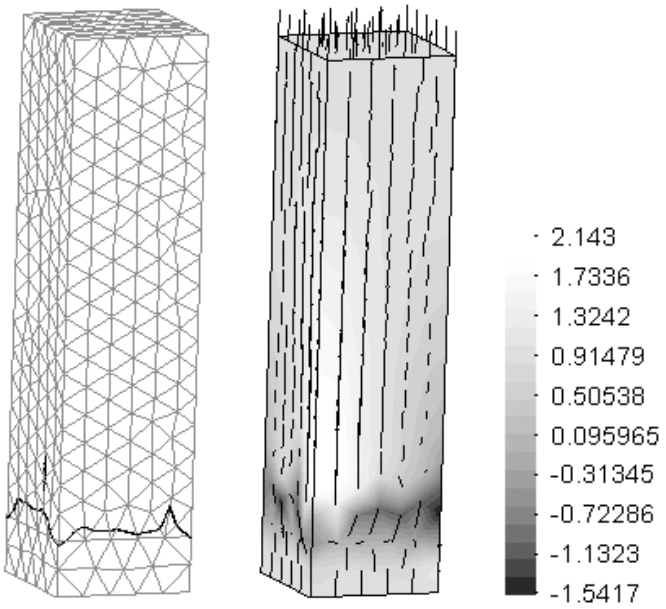


Figure 2.7: Interface shape and vertical velocity band plot together with velocity vectors using only improved integration

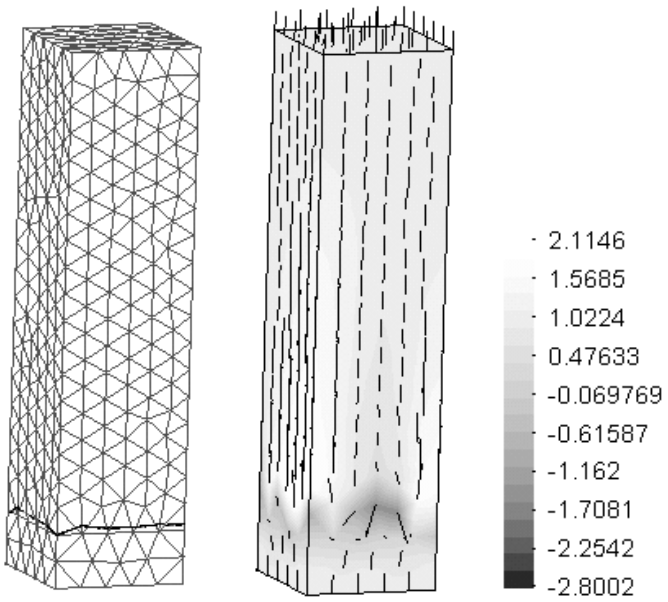


Figure 2.8: Interface shape and vertical velocity band plot together with velocity vectors using jump properties and no enrichment nor modified integration

In Figures 2.6, 2.7, 2.8 and 2.9 we show the shape of the interface and the velocity field for four different cases. In Fig. 2.6 the results with pressure enrichment are shown. The interface remains flat as expected and its displacement corresponds to the amount of injected fluid. When a modified integration but no pressure enrichment is used (see Fig. 2.7) the results are very poor. The velocity shows important oscillations close to the interface and there is an important mass loss. The mass loss is so important that the free surface remains nearly at its initial height. Without using pressure enrichment nor modified integration and approximating the material properties with the option described as *Heaviside* properties, the results (shown in Fig. 2.8) are as bad the those described for the previous case. Finally, when *pseudo-concentration* properties are used (see Fig. 2.9), without pressure enrichment nor modified integration, the mass conservation improves with respect to the previous two cases, but is worse than that obtained with the formulation proposed in this chapter. As in the other two cases without pressure enrichment, the errors in the prediction of the vertical velocity close to the interface can

be twice the inlet velocity. The source of these errors is the impossibility of the shape functions to capture the discontinuous pressure gradient that exists at the interface. When the pressure is enriched the velocity errors nearly disappear. The slight errors that remain can be attributed to the fact that the numerical resolution of the level set is not exact and therefore the deviations in the shape of the interface give rise to small variations in the velocity.

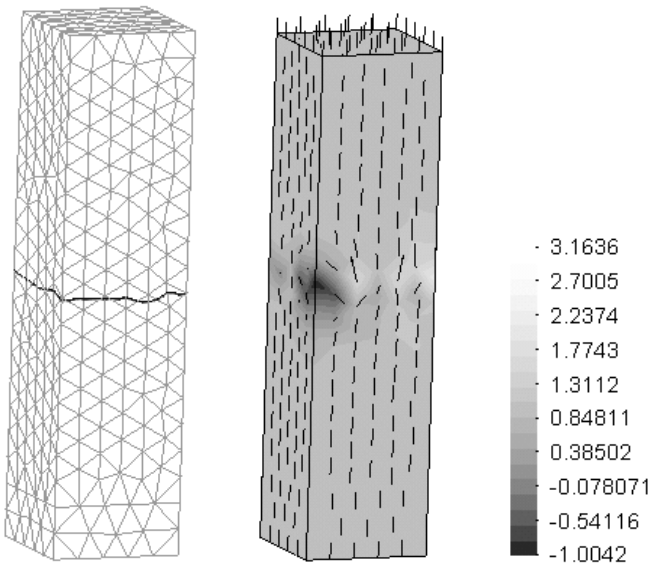


Figure 2.9: Interface shape and vertical velocity band plot together with velocity vectors using variable properties and no enrichment nor modified integration

2.2.3 Sloshing problem

As a final example, we consider a sloshing problem. It is a simple problem of free oscillation of an incompressible liquid in a container. Following [99], [100], and [98] we consider a liquid column of width b with an initial surface profile corresponding to the first antisymmetric mode of vibration. The height of the free interface is

$$\eta\left(x,0\right)=1.0+a\ \sin\frac{\pi}{b}x$$

where a is the amplitude of oscillation. Thus, the initial condition for the level set function is

$$\psi(x, y, 0) = \eta(x, 0) - y$$

where $x = 0$ at the middle of the container and $y = 0$ at the bottom. Using the same parameters as in the cited papers, the amplitude is taken as $a = 0.01$ and the kinematic viscosity as $\nu = 0.01$. The previous papers use a Lagrangian formulation and therefore only one fluid is solved and the computational domain is allowed to deform. Since we use an Eulerian formulation the computational domain does not deform but a second fluid is also simulated. The density and dynamic viscosity of the second fluid are 100 times smaller than those of the bottom one so that it does not affect significantly the flow of the heavier fluid. The container walls are assumed to be impermeable and allow for free slip. The domain we use has a width $b = 1$ and a height $h = 2$ and is filled with equal amounts of each fluid (see Fig. 2.10). *Heaviside* properties are used to approximate the material properties in elements cut by the interface. The mesh has 1394 triangular elements and 747 nodes and is refined close to the interface.

In Fig. 2.11 we show the position of the interface after 11 seconds, using: (1) enriched pressures, (2) only improved integration and (3) no modification. Figure 2.12 shows the computed time history of $\eta\left(\frac{b}{2}, t\right)$ for the three cases together with the results presented by Ramaswamy et al. [100] Both figures show that in the cases without pressure enrichment there is a significant mass loss. The enriched simulation agrees closely with the results reported by Ramaswamy et al. [100]

2.3 Conclusions

In this chapter we have presented an enrichment for the pressure finite element shape functions that allows to improve the solution of two phase flows. The benefits introduced by the method show up clearly as the gravitational forces increase. The Froude number

$$Fr = \frac{U^2}{gL},$$

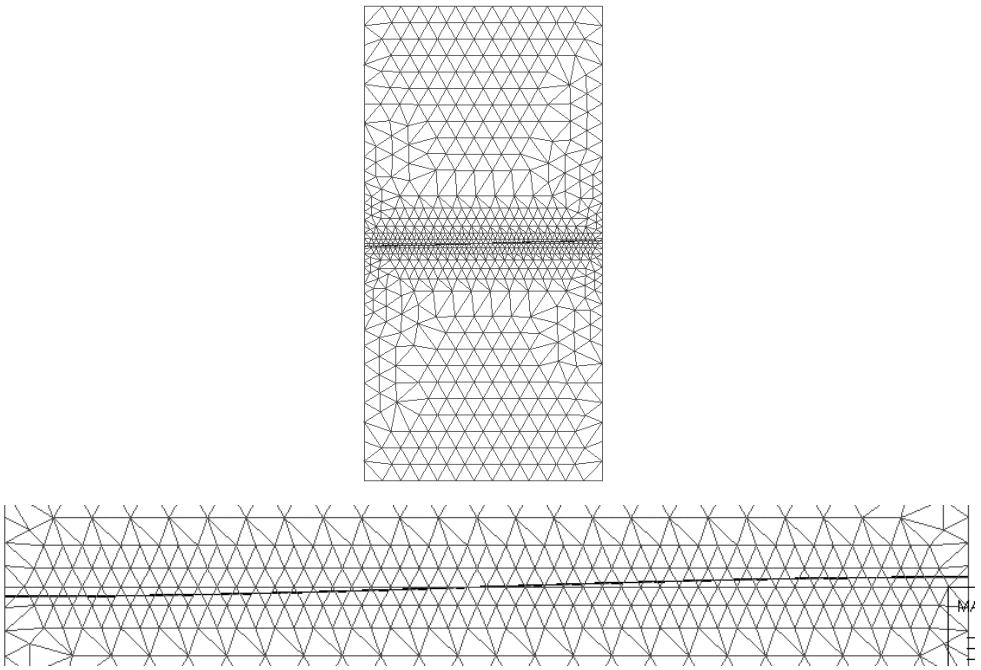


Figure 2.10: Mesh and initial interface position for the sloshing problem. Top: general view. Bottom: detail.

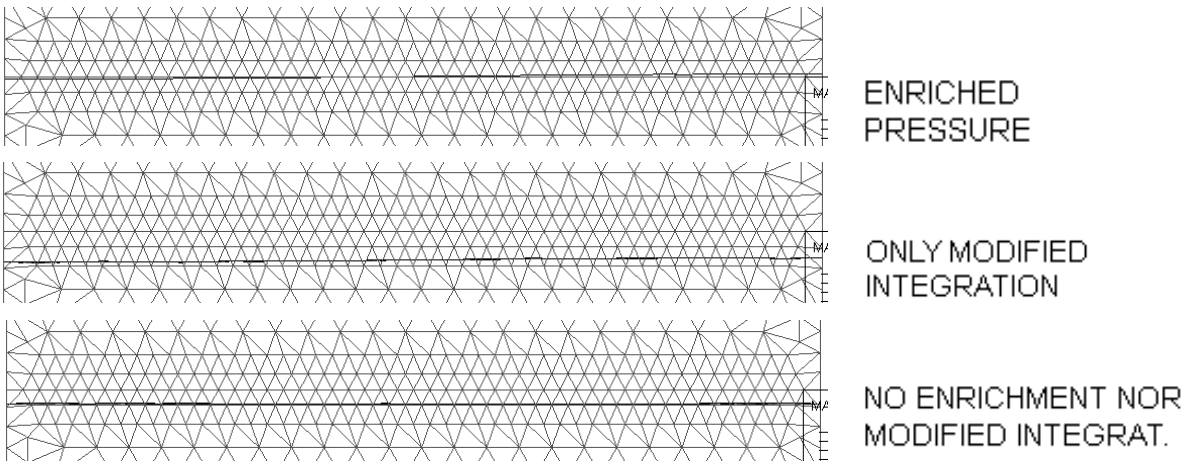


Figure 2.11: Interface position at $t = 11$s for the sloshing problem

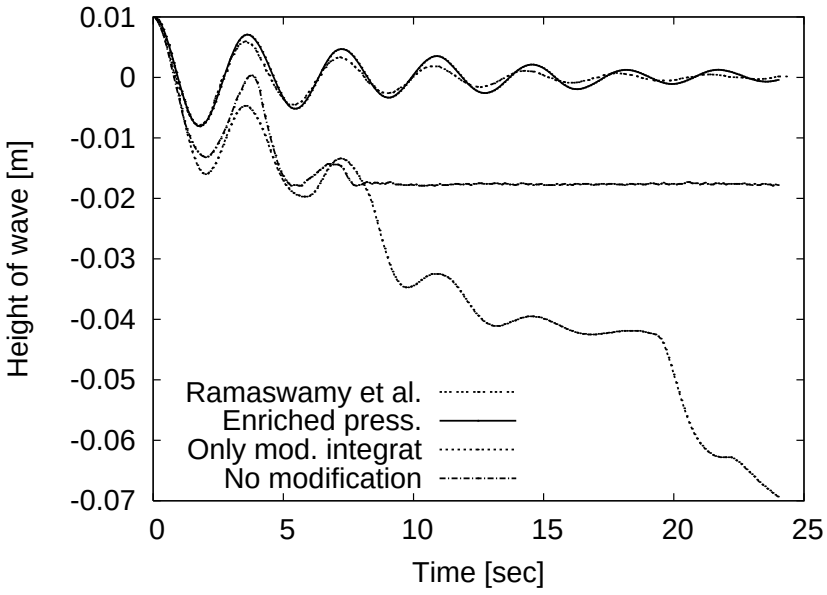


Figure 2.12: Time histories of surface elevation amplitude for the sloshing problem

characterizes the ratio of the inertial to the gravitational forces in free-surface flows, where U is a characteristic velocity, g is the value of the gravitational acceleration and L is a characteristic length. A modified Froude number can be used for two fluid flows

$$Fr_m = \frac{U^2}{gL}\frac{\rho_1}{(\rho_1 - \rho_2)},$$

where ρ_1 is the density of the heavier fluid and ρ_2 is the density of the lighter one. The errors without using the enrichment increase as the Froude number tends to zero. As the gravitational forces tend to dominate the flow, small errors in the pressure can give rise to big errors in the velocities. Therefore the type of enrichment presented in this paper is specially useful for low modified Froude number flows. The enrichment used is local to each element cut by the interface and can therefore be condensed prior to assembly, making the implementation quite simple on any finite element code.

It is interesting to note that, at least for the problems presented in this paper, the proposed solution can reduce the errors by one or two orders of magnitude (see vertical tube results) and thus make problems solvable with quite coarse meshes.

Despite we have not presented numerical results for the increase in CPU time for

obtaining the system to be solved per iteration introduced by the enrichment and improved integration, we have observed that it is generally less or much less than 10 percent. Taking into account that the method allows to reduce the number of iterations per time step and also to use coarser meshes the mentioned CPU time increase is by far counter balanced.

We have not presented results for the pressure because it is difficult to do so with a typical post-processing program in the enriched elements, but it is evident from the improvements shown for the velocities and shape of the interface that the pressure, since it is the only unknown we have modified, must have also improved.

The examples shown in this paper demonstrate that the proposed enrichment can introduce significant improvements. It allows to avoid spurious velocities and enhances mass conservation.

Chapter 3

A free surface model

Free surface flows are a special kind of flows with moving interface where the influence of one of the fluids over the other one is negligible. As discussed in Chapter 1, CFD approaches for moving interfaces problems are typically categorized into two main groups: Eulerian, fixed mesh or interface capturing techniques [18, 31, 57, 73, 90, 106, 110] and Lagrangian, and in the more general case Arbitrary Lagrangian Eulerian (ALE), moving mesh or interface tracking techniques [98–100]. The model we will propose is clearly a fixed mesh, interface capturing technique but it is not so obvious wether to classify it as an Eulerian or as a Lagrangian formulation. Actually the model we propose has two versions: a simplified one which solves the momentum equations in an Eulerian manner and another one that uses an ALE formulation but on a fixed mesh.

As we have mentioned previously, in interface capturing techniques a fixed computational domain is used together with an interface function to capture the position of the interface. The interface is captured within the resolution of the fixed mesh and the boundary conditions at the interface are somehow approximated. In interface tracking techniques the mesh is updated in order to track the interface. The simplest approach is to deform the mesh without changing its topology, but this is possible only for very simple flows. As the flow becomes more complex and unsteady, remeshing is required, and consequently the projection of the results from the old to the new mesh are needed. In 3D calculations, these operations can introduce costs that can render moving

mesh techniques unfeasible. This is the main reason why we prefer to use fixed mesh methods. Both approaches can deal with two phase or free surface flows. In moving mesh simulations [71, 87, 98] it is very common to see real free surface simulations where only one fluid is modelled and the mesh follows the movement of the interface. On the other hand, in finite element fixed mesh simulations free surface flows are typically treated as a particular case of two phase flows where the second fluid is air or some pseudo fluid with a density and viscosity much smaller than the first fluid.

The typical Eulerian approach can work well in a great number of situations but in some cases it can fail miserably. For example, flow of different density fluids under the action of gravity. This problem has been analyzed in the previous chapter (see also [36, 37]). The main problem is that since we are using a fixed mesh some elements are cut by the interface, where the pressure gradient is discontinuous. Such a pressure field cannot be accurately represented by the finite element shape functions and if the two most important terms in the momentum equation are the pressure gradient and the gravitational forces, huge errors can be introduced in the velocity field that can spoil the simulation. In the previous chapter we proposed a solution based on enriching the pressure shape functions in the elements cut by the interface that is valid for two phase flows. In this chapter we will propose an alternative solution that is valid only for free surface flows. This might seem a disadvantage with respect to the previous method that can deal with the more general case of two phase flows. We nevertheless believe that a solution that takes advantage of the particularities of free surface flows deserves special attention due to the fact that a great proportion of two phase flows of practical interest are free surface flows. Since the effect of surface tension is negligible in the flows we are interested in we will not take such effects into account in the model we propose.

The solution we propose in this paper is to model free surface flows on fixed grids in a very similar way to the one used for two phase flows but modelling in principle only the part of the domain filled by the liquid. Since we are using a fixed grid method we will have elements that are totally filled by liquid and others only partially. The main question is what to do in partially filled elements. What we propose to do is to integrate

only in the part filled by the fluid. In order to do so we use special integration rules in elements cut by the front that have been introduced in the previous chapter. This will allow us to impose the correct boundary conditions on the free surface, the key to the success of the method.

As we have already mentioned, fixed mesh methods generally share two basic steps. In the first one, the motion in both phases is found as the solution of the Navier–Stokes equations for one phase flow with variable properties. In the second one, an equation for an interface function that allows to determine the position of the interface, and thus the properties to be assigned in the previous step, is solved. The model we will present has two versions, one that solves the Navier–Stokes equations in an Eulerian manner and the other one that uses an ALE formulation. In both versions a level set function is used to determine the position of the interface.

In the next section we introduce the ALE formulation that is used by one of versions of the model we propose. Obviously when the domain velocity is zero the ALE formulation reduces to the Eulerian description presented in Chapter 1. In the next Section we present the free surface model that uses the fixed mesh ALE formulation. In the third Section a simplified Eulerian version of the free surface model is presented. In the fourth Section we analyze three numerical examples that have already been used in Chapter 2. We demonstrate that the free surface model, in any of its two versions, allows to obtain results that are similar or better to the ones obtained with enriched pressure shape functions and much better than those obtained with a typical two phase Eulerian model.

Most of the work we will describe in this Chapter has been published in [38]. Further numerical examples have been included in [39].

3.1 ALE description of the Navier–Stokes equations

The velocity and pressure fields of an incompressible fluid in a moving domain Ω during the time interval (t_0, t_f) can be described by the incompressible Navier–Stokes equations

in the ALE form:

$$\rho\Big[\frac{\partial \boldsymbol{u}}{\partial t} + ((\boldsymbol{u}-\boldsymbol{u}_d)\cdot\nabla)\boldsymbol{u}\Big] - \nabla\cdot[2\mu\boldsymbol{\varepsilon}(\boldsymbol{u})] + \nabla p = \boldsymbol{f}, \tag{3.1}$$

$$\nabla\cdot\boldsymbol{u} = 0, \tag{3.2}$$

where $\boldsymbol{u}_d$ is the domain velocity. The boundary and initial conditions are the same as introduced in Chapter 1. In [29] we present a detailed derivation of the ALE method for the Navier–Stokes equations.

Following the steps described in Chapter 1 the problem can be discretized both in space and time. The ASGS monolithic discrete problem associated with the Navier–Stokes equations (3.1)-(3.2), discretizing in time using the generalized trapezoidal rule, and linearizing the convective term using a Picard scheme, can be written as follows: Given a velocity $\boldsymbol{u}_h^n$ at time t^n, a domain velocity $\boldsymbol{u}_{d,h}^{n+\theta}$ at time $t^{n+\theta}$, and a guess for the unknowns at an iteration $i-1$ at time t^{n+1}, find $\boldsymbol{u}_h^{n+\theta,i} \in \boldsymbol{V}_{h,u}$ and $p_h^{n+\theta,i} \in Q_h$, by solving the discrete variational problem:

$$\begin{aligned}
&\int_\Omega \rho\frac{\boldsymbol{u}_h^{n+\theta,i}-\boldsymbol{u}_h^n}{\theta\delta t}\cdot\boldsymbol{v}_h\,\mathrm{d}\Omega + \int_\Omega \rho((\boldsymbol{u}_h^{n+\theta,i-1}-\boldsymbol{u}_{d,h}^{n+\theta})\cdot\nabla)\boldsymbol{u}_h^{n+\theta,i}\cdot\boldsymbol{v}_h\,\mathrm{d}\Omega \\
&\quad + \int_\Omega 2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}_h^{n+\theta,i}) : \boldsymbol{\varepsilon}(\boldsymbol{v}_h)\,\mathrm{d}\Omega - \int_\Omega \nabla\cdot\boldsymbol{v}_h p_h^{n+\theta,i}\,\mathrm{d}\Omega - \int_\Omega \boldsymbol{v}_h\cdot\boldsymbol{f}\,\mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e} \tau_1^{n+\theta,i-1}\Big[\frac{\mu}{\rho}\Delta\boldsymbol{v}_h + ((\boldsymbol{u}_h^{n+\theta,i-1}-\boldsymbol{u}_{d,h}^{n+\theta})\cdot\nabla)\boldsymbol{v}_h\Big]\cdot\Big[\frac{\rho}{\theta\delta t}(\boldsymbol{u}_h^{n+\theta,i}-\boldsymbol{u}_h^n) \\
&\qquad - \nabla\cdot[2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}_h^{n+\theta,i})] + \rho((\boldsymbol{u}_h^{n+\theta,i-1}-\boldsymbol{u}_{d,h}^{n+\theta})\cdot\nabla)\boldsymbol{u}_h^{n+\theta,i} + \nabla p_h^{n+\theta,i} - \boldsymbol{f}\Big]\,\mathrm{d}\Omega \\
&\quad + \sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e} \tau_2^{n+\theta,i-1}(\nabla\cdot\boldsymbol{v}_h)(\nabla\cdot\boldsymbol{u}_h^{n+\theta,i})\,\mathrm{d}\Omega = 0, \\
&\int_\Omega \rho q_h\nabla\cdot\boldsymbol{u}_h^{n+\theta,i} + \sum_{e=1}^{n_{\mathrm{el}}}\int_{\Omega^e}\tau_1^{n+\theta,i}\nabla q_h\cdot\Big[\frac{\rho}{\theta\delta t}(\boldsymbol{u}_h^{n+\theta,i}-\boldsymbol{u}_h^n) - \nabla\cdot[2\mu\boldsymbol{\varepsilon}(\boldsymbol{u}_h^{n+\theta,i})] \\
&\qquad + \rho((\boldsymbol{u}_h^{n+\theta,i-1}-\boldsymbol{u}_{d,h}^{n+\theta})\cdot\nabla)\boldsymbol{u}_h^{n+\theta,i} + \nabla p_h^{n+\theta,i} - \boldsymbol{f}\Big]\,\mathrm{d}\Omega = 0\ ,
\end{aligned}$$

for $i = 1, 2, ...$ until convergence, that is to say, until $\boldsymbol{u}_h^{n+\theta,i} \approx \boldsymbol{u}_h^{n+\theta,i-1}$ and $p_h^{n+\theta,i} \approx p_h^{n+\theta,i-1}$ in the norm defined by the user.

In the ALE case the parameters τ_1 and τ_2 depend on the ALE velocity $\boldsymbol{u}_a = \boldsymbol{u} - \boldsymbol{u}_d$:

$$\tau_1 = \frac{\rho(h^e)^2}{4\mu + 2\rho h^e|\boldsymbol{u}_a^e|}\ , \qquad\qquad \tau_2 = 4\mu + 2\rho h^e|\boldsymbol{u}_a^e|\ .$$

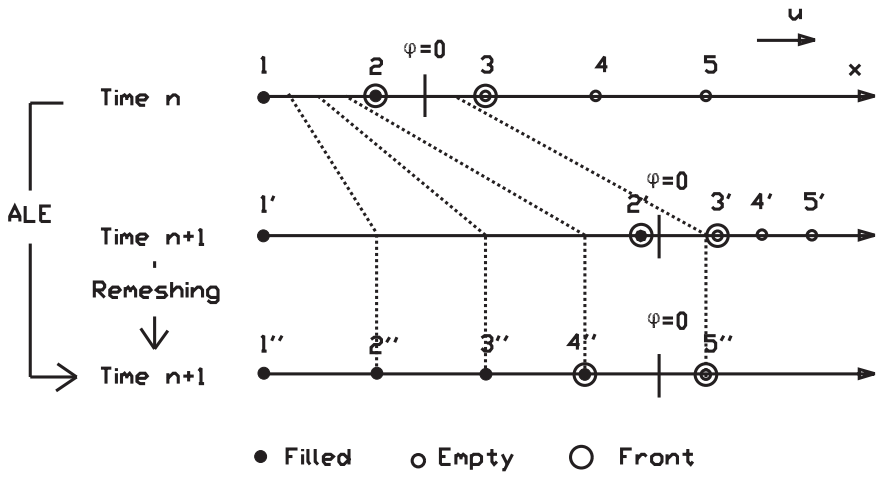
Once the algorithm has produced a converged solution, the velocity field at t^{n+1} can be updated from the velocity at $t^{n+\theta}$ by using the relation $\boldsymbol{u}^{n+1} = [\boldsymbol{u}^{n+\theta} - (1-\theta)\boldsymbol{u}^n]/\theta$.

3.2 FM-ALE free surface model

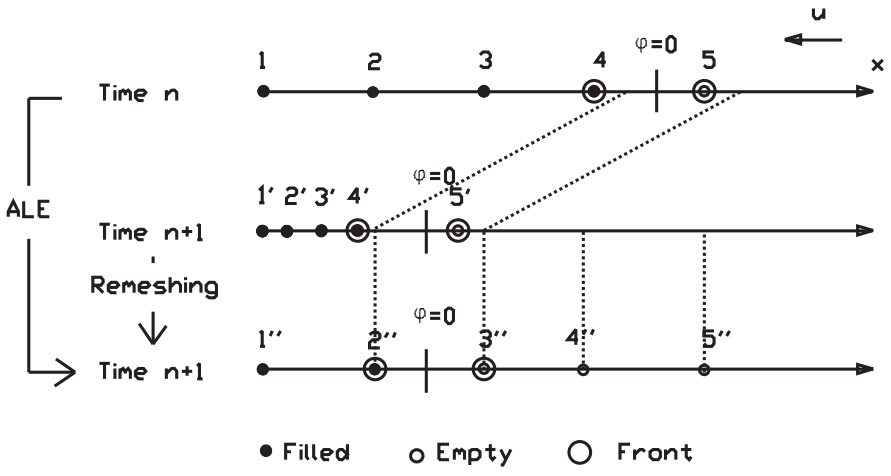
In this section we present a new free surface model on fixed meshes. We will start from a typical Eulerian simulation for a two phase flow and describe the way in which our model departs from it.

As it has already been mentioned, a typical Eulerian simulation includes two main steps. One which solves Eulerian two fluid Navier–Stokes equations and the other one which determines the interface position. Both are solved over the entire mesh. When free surface flow is considered, the properties used in the second fluid are much smaller than those in the main fluid. The model we propose solves the Navier–Stokes equations only on one fluid bounded by a moving free surface whose position, as in the typical model, is determined by the level set function. Therefore the domain Ω where we solve the Navier–Stokes equations does not extend over the whole mesh, but only over totally filled elements and over the filled part of elements cut by the interface. This is an important difference with typical finite element simulations (Eulerian, Lagrangian or ALE) where the domain that is simulated extends over the whole mesh. In order to be able to use a domain that includes portions of elements, special integration rules have to be used. The integration rule we use here has been presented in the previous chapter and consist in dividing the elements cut by the front into sub elements only for integration purposes. In a finite element setting the free surface boundary condition is a natural boundary condition, and by using enhanced integration we are able to impose it correctly even if the finite element faces do not coincide with the interface. The possibility of imposing the correct boundary conditions on the interface without having to resort to a moving mesh formulation is one of the key assets of the method. It allows us to take advantage of the best of both worlds (Eulerian and Lagrangian). If instead of $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{0}$, we would like to impose some prescribed value $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{t}$ on the free surface, the procedure would be

slightly more complicated. Enhanced surface integration rules would have be to defined. The idea could be very similar to the enhanced volume integration used to integrate only on the filled part of cut elements. In each cut element, the front (as defined by the level set) would be divided into triangles where the usual surface integration rules would be used. Thus one would be able to build the term corresponding to the Neumann boundary condition applied on the interface, $\int_\Gamma \boldsymbol{v} \cdot \boldsymbol{t} \, \mathrm{d}\Gamma$.



(a) positive velocity



(b) negative velocity

Figure 3.1: One dimensional FM-ALE example

The second main difference with typical fixed mesh interface simulations is that we

use an ALE description when solving the Navier–Stokes equations. The particularity of our ALE description is that it is used on a fixed mesh. It is called FM-ALE and has been introduced in a paper on lost foam casting [59]. In this chapter we extend it to free surface flows. The idea behind the FM-ALE method is quite simple; if one wants to simulate a moving domain on a fixed grid using an ALE description what one must do is to project the results obtained on the deformed domain on the portion of the fixed mesh occupied by the fluid at each time step.

The method we propose consist of three main steps in going from time n (where velocities and interface position are given) to time $n+1$:

1) Find the interface position at step $n+1$ by solving the level set equation. The velocities obtained at step n are used in this step.

2) Obtain the domain velocity and fluid velocity at time n on the new domain determined in the previous step.

3) Solve the Navier-Stokes equations on the new domain using an ALE description.

The way in which to solve the first and third steps has already been discussed. We will now describe the way in which to deal with the second step. The key point here is how to define the domain velocity at time $n+1$. It will depend on the fluid velocity at time n and on the interface position at times n and $n+1$. We will use the one dimensional example shown in Figure 3.1 to explain how it is obtained. Contrary to what happened in the original FM-ALE paper [59] where the domain could only expand, in a free surface simulation the domain can either expand or contract.

The domain velocity at time $n+1$ will depend on the fluid velocity at time n and on the level set function at times n and $n+1$. In order to describe the method we will classify the nodes into empty and filled nodes. The nodes belonging to cut elements will be additionally described as front nodes. The domain velocity will be set to zero at nodes that belong to the fluid but not to the front at times $n+1$ and n on the fixed mesh. That is the case of node 1 in Figures 3.1a and 3.1b, where Figure 3.1a represents the case where the domain has a positive x velocity (it is expanding) and Figure 3.1b represents the case where the domain has a negative x velocity (it is contracting). The domain velocity will

be set to

$$\boldsymbol{u}_d^{n+1} = \boldsymbol{n}(\boldsymbol{u}^n \cdot \boldsymbol{n}),$$

at nodes that belong to the front or to the air at time n. The normal direction $\boldsymbol{n}$ is obtained from the level set function at each node as

$$\boldsymbol{n} = \frac{\nabla\psi^{n+1}}{|\nabla\psi^{n+1}|},$$

Such is the case of nodes 2, 3, 4 and 5 at Figure 3.1a and nodes 4 and 5 at Figure 3.1b. Using the previous conventions, in the expanding case (Figure 3.1a) the domain velocity is defined on all the nodes. In the contracting case it is not be defined at nodes 2 and 3. There it is obtained by solving

$$\Delta\boldsymbol{u}_d = 0,$$

with the boundary conditions defined previously. Once the domain velocities have been obtained, the displacements in one time step are simply $\delta\boldsymbol{x} = \boldsymbol{u}_d\delta t$. They allow us to obtain the deformed mesh, shown in the middle line. Finally the mesh velocities and fluid velocities at time n can be obtained on the fixed mesh at time $n+1$ (bottom line) by interpolating (or by projecting) on the deformed mesh as shown in dotted lines. These two values are the ones needed to solve the Navier–Stokes equations in ALE form.

Following our presentation in [29] the second step can be divided into four substeps:

- *Virtually* deform the mesh at time n (M^n) to a virtual mesh at time $n+1$ (M_{virt}^{n+1}) using classical ALE concepts and compute the mesh velocity $\boldsymbol{u}_d^{n+1}$.
- Write down the ALE Navier Stokes equations on M_{virt}^{n+1}.
- Define a new mesh at time $n+1$ (M^{n+1}) formed by totally filled elements of the background mesh and the filled part of cut elements.
- Project the ALE Navier Stokes equations from M_{virt}^{n+1} to M^{n+1}.

How we compute the mesh velocity $\boldsymbol{u}_d^{n+1}$ has already been described. This defines the displacements and therefore M_{virt}^{n+1} can be obtained. M_{virt}^{n+1} is shown in blue in the

schematic of the FM-ALE approach presented in Figure 3.2. Writing down the ALE Navier Stokes equations on M_{virt}^{n+1} (see [29]) leads to a system of equations that is not actually formed when the FM-ALE approach is used. Instead the ALE Navier Stokes equations are projected onto to M^{n+1}. For the definition of the new mesh at time $n+1$, free surface problems involve Neumann boundary conditions on the interface that can be easily imposed when enhanced integration is used in the elements cut by the front as we have already mentioned. For other problems, that require Dirichlet conditions on the interface, we mention two possibilities in [29]; adding new nodes or imposing boundary conditions approximately.

Projecting the ALE Navier Stokes equations from M_{virt}^{n+1} to M^{n+1} implies projecting both $\boldsymbol{u}_{virt}^{n}$ and $\boldsymbol{u}_{d,virt}^{n+1}$ from M_{virt}^{n+1} to M^{n+1}. It is important to stress that, as it is well known in the classical ALE approach, $\boldsymbol{u}^{n}$ is known on M_{virt}^{n+1} because the nodes of this mesh are obtained from the motion of the nodes of M^{n} with the mesh velocity $\boldsymbol{u}_{d,virt}^{n+1}$. The projection of $\boldsymbol{u}_{virt}^{n}$ onto M^{n+1} clarifies the effect of the mesh motion in the context of fixed mesh methods. In particular, there is no doubt about the velocity at previous time steps of newly created nodes. Since the mesh velocity is computed on M_{virt}^{n+1} it also needs to be projected to compute on M^{n+1}.

It is interesting to note that p^{n+1} is not the projection of p_{virt}^{n+1} onto M^{n+1}. Pressure p^{n+1} is determined by imposing that $\boldsymbol{u}^{n+1}$ is divergence free, which at the discrete level is not equivalent to impose that $\boldsymbol{u}_{virt}^{n+1}$ is divergence free.

We will now go back to describe a particularity of the method, concerning the solution of the level set equation. We have already mentioned the level set equation is solved over the whole mesh and therefore the velocities on the region formed by non filled elements must be defined in some way. As we have mentioned in Chapter 1, this is not a big problem, since in theory the only velocities needed to transport the interface position are those on the interface defined by $\psi = 0$. Several options to define extension velocities that allow us to obtain a velocity field on the whole mesh from the velocities on the interface can be found in the literature [90]. The approach we will use is a relatively simple one [59]: on the part of the mesh formed by empty elements a stationary Stokes problem will be

solved. On the air front nodes the velocities obtained when solving the fluid will be used as Dirichlet boundary conditions. The boundary conditions used when solving for the extension velocities have little influence on the resulting fluid flow and will be taken as in a typical Eulerian two fluid simulation. The fluid properties used when solving the Stokes problem in the empty region are those of the fluid. The advantages of the method used to obtain the extension velocities is that they are divergence free and satisfy boundary conditions.

From the description of the FM-ALE formulation it is seen that the major differences with respect to the classical ALE approach are the following:

- Given a position of the fluid front on the fixed mesh, elements cut by the front are split into subelements (only for integration purposes), so that the front coincides with the edges of the subelements. This allows to prescribe $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \mathbf{0}$ as boundary condition. In fact, it is only necessary to modify the integration rule, as explained earlier. Once this is done, the flow equations can be solved.

- After deforming the mesh from one time step to the other using classical ALE procedures, results are projected back to the original mesh (through interpolation, projection with restrictions as explained in [58] or any other technique).

- The definition of the fluid front is represented by the level set function, and not by the position of the material points at the free surface as in a classical ALE method. Note that the previous step implies that front nodes cannot be tracked.

A schematic of the FM-ALE approach for free surface flows is presented in Figure 3.2.

In essence, this formulation is the same as the one presented in [59] with two major differences:

- In [59], where the fluid domain always expands, it was enough to move the nodes adjacent to the front. In the present model, since the domain can also contract, a more general definition for the mesh velocities had to be adopted. The mesh velocities at time $n+1$ depend on the fluid velocities at time n and on the level set function both at times n and $n+1$. When the domain contracts some nodes of the fluid domain are moved by solving equation (3.2) with boundary conditions determined from the rules used to define

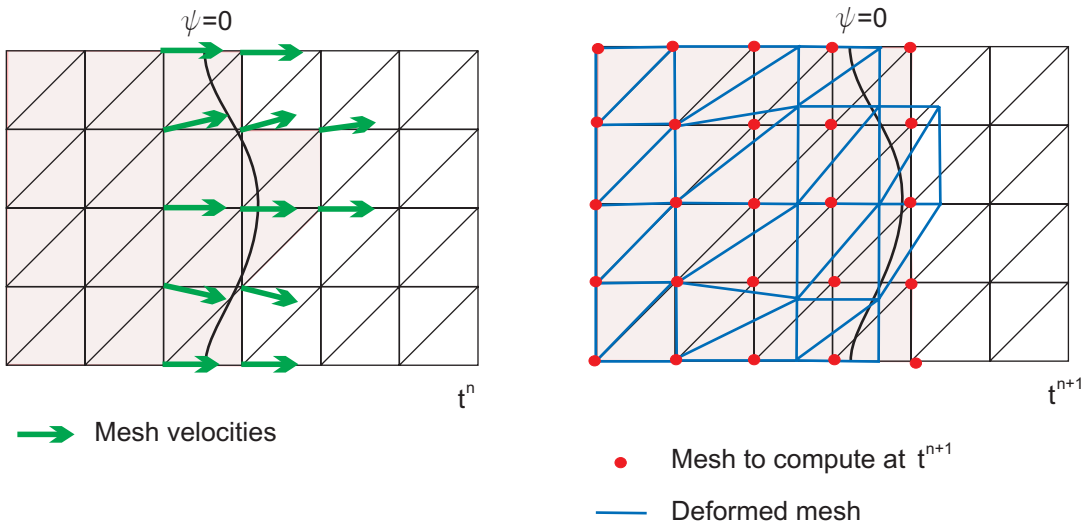


Figure 3.2: Two dimensional FM-ALE schematic

the mesh velocity on the rest of the nodes.

- In the lost foam model of [59], the front velocity is given and the pressure is unknown. In the present free surface model, the front velocity is unknown and the traction is given ($\boldsymbol{n} \cdot \boldsymbol{\sigma} = \mathbf{0}$).

3.3 Eulerian simplified free surface model

In the previous section we have presented a version of the free surface model that solves the Navier–Stokes equations on the fluid only using an ALE description. In this Section we will introduce a simpler version, the main difference being that an Eulerian description is used. The model will therefore be more closely related to the typical Eulerian two phase model. Nevertheless the key element of the model, solving the Navier–Stokes equations only in the fluid region Ω (totally filled elements plus filled part of cut elements), will remain unaltered.

The way to understand this model is to think of it as a typical two phase flow model, with a second fluid with negligible properties and where the null traction boundary condition has been added on the interface. This boundary condition allows us to uncouple

the solution of the fluid from that of the air making the problem much simpler. We will often refer to the second fluid as air even if it may be some other fluid. Suppose that we are solving a typical two phase flow model using enhanced integration in the elements cut by the front. When one looks at the discrete momentum equation it is clear that the contribution to the matrix and right hand side corresponding to the transient, convective, diffusive and external force terms will tend to zero in the air elements and in the part of the cut elements filled by air. Thus all the terms in the momentum equation, except for the pressure gradient, tend to the same value if one uses the free surface model we propose or a two phase model with a negligible second fluid. In order to interpret what happens with the pressure term when using the free surface methodology as compared to the two phase case, one has to observe that if all other terms tend to zero in the region not occupied by the liquid then the solution there would have a null pressure gradient. If one introduced such information a priori in the two phase case, then the systems resulting from the momentum equation would be identical when the fluid properties tend to zero. The problem with the two phase model, when used with the typical finite element functions, is that they cannot represent a pressure gradient that is zero in one part of the element and different from zero in the other. As we have already said, since the velocity and pressure are coupled, the impossibility of accurately representing the pressure can introduce errors in the velocity that can render the solution meaningless. Enriching the pressure shape functions is a way to solve such problem [37], the free surface formulation we present here is another solution, perhaps simpler. They will be compared numerically in Section 5.4.

Finally one also has to look at the continuity equation. The contribution to the system matrix would be different if one uses equation (1.2). But it could be replaced by

$$\rho \nabla \cdot \boldsymbol{u} = 0. \tag{3.3}$$

From the continuum point of view both equations are equivalent. The numerical approximation of equation (3.3) would lead to identical system matrices if one uses the free surface model or the two phase model. Using equation (3.3) can be seen as weighting the incompressibility constraint depending on the density of the fluid. In the free surface

case what we are doing is only imposing incompressibility in the region where fluid exists. It is a pretty logical hypothesis.

One important point to remark is that as we are using enhanced integration we are able to integrate only in the domain filled by liquid and thus we can impose the Neumann boundary condition corresponding to the free surface ($\boldsymbol{n}\cdot\boldsymbol{\sigma} = \mathbf{0}$) accurately exactly where the interface is located according to the level set function. This is a key point for the success of the method. On the other hand, since we are only simulating one fluid we do not have a discontinuous pressure gradient and therefore no special shape functions are needed.

As in the FM-ALE free surface model described in the previous Section, the level set equation must be solved on the whole mesh, not only in the liquid region. In order to reduce computational labor the level set equation could be solved only in a small region close to the interface using the narrow-band approach [104]. Once again, the velocity must be defined in the empty region somehow, in order to transport the level set function. In this case the solution adopted is very similar to the one used in the previous Section except for the fact that instead of solving the steady Stokes equations with fluid properties we will use the transient Navier-Stokes equations with air (or some pseudo fluid) properties. The boundary conditions will be the same as in the FM-ALE model. The reason for this choice is that we are justifying this model based on what happens in the Eulerian two phase case and therefore it seems logical to solve for the air as similarly as possible as done in that model. We would nevertheless like to point out that the way in which the air is solved is not of great importance.

Contrary to what happens in the FM-ALE model where the velocities in the empty region are only needed to transport the level set, in the Eulerian free surface model there is another reason for obtaining those velocities. Since the fluid domain is moving and the mesh is fixed there will be nodes that in step $n+1$ belong to the fluid but in the previous step belonged to the empty region. In such nodes we will use the velocities calculated for the empty region when modeling the transient term. The validity of such approach is justified once again pointing out that it is with what happens in the case of a two phase

flow when the properties of the second fluid tend to zero.

3.4 Numerical examples

In this section we present three numerical examples where one can appreciate the benefits the proposed formulation can provide compared to a typical two-phase flow model applied to free surface flow. The examples are almost the same ones used in Chapter 2. The results obtained with the free surface model will be compared with the results obtained with a typical two-phase flow model and also with the enriched pressure model presented in the Chapter 2.

3.4.1 Two–fluid cavity

The first example, called the two–fluid cavity, is a square domain filled with equal amounts of two different density fluids and a fixed horizontal velocity ($0.1\ m/s$) on the bottom wall. In the free surface case only the lower half of the mesh is filled by fluid. The walls are supposed frictionless and the top face is left open. It is a very simple example but it can be representative of the numerical problems that can appear in much more complex problems.

The same three unstructured triangular meshes as in Chapter 2 were used (see Table 2.1) . The square domain has a side length $L = 10\ m$. The material properties used (SI units) are $\rho_1 = 1000$, $\mu_1 = 10$ for the fluid on the bottom. The viscosity of the bottom fluid is 1000 times the viscosity of water so as to obtain a relatively low Reynolds number ($Re = 100$) in order to avoid unnecessary complications. The simulations where run for 100 seconds with a 0.5 second time step size. In the two phase flow simulations the properties of the second fluid used are 100 times smaller than those of the first fluid so as to be in a free surface case. They differ from those used in the previous Chapter where they were similar to the properties of the first fluid because now we want the second fluid not to influence the first one. With such density difference the problem turned out to be more difficult to solve and therefore for this example the acceleration of gravity is reduced

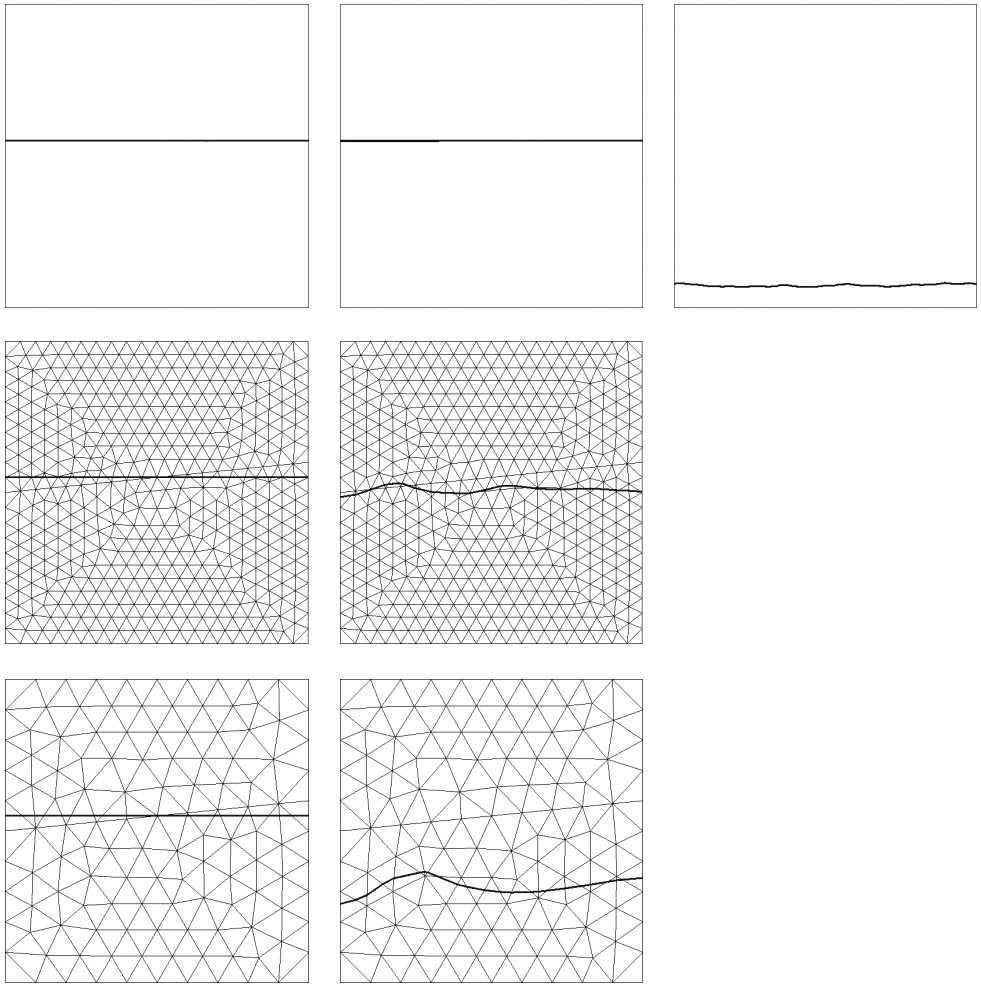


Figure 3.3: Shape of the interface for the different meshes and numerical models at $t = 100$ s

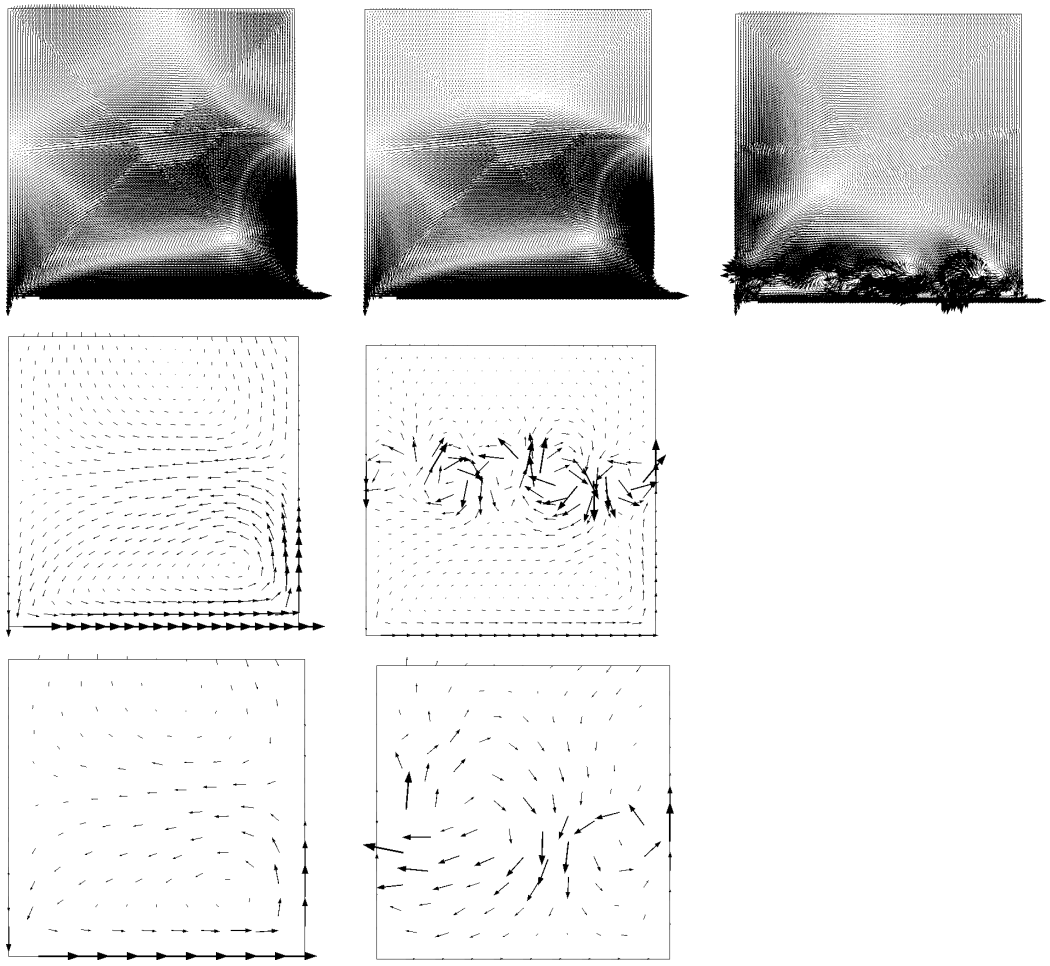


Figure 3.4: Flow pattern for the different meshes and numerical models at $t = 100$ s

to $g = 1.0$. In the rest of the examples it is $g = 10.0$.

In Fig. 3.3 we show the shape of the interface for the three meshes with three different models: The FM-ALE free surface model, the enriched pressure two–phase model (Chapter 2) and the normal two–phase model.

Using the finest mesh the free surface model and the enriched pressure two–phase model give nearly the same result. The normal two–phase model fails to predict the correct interface position. There is an erroneous fall of the interface created by an incorrect velocity field as is shown in Fig. 3.4. Despite that we have not got physical measurements, the two correct solutions obtained with this mesh can be taken as a reference against which we can compare the results obtained with the other meshes. Using the medium mesh, only the free surface model attains results as good as those obtained with the fine mesh. The enriched pressure two–phase model produces some distortion of the interface. Using the normal two phase model the solution obtained on the medium and coarse meshes is so bad that the interface has disappeared from the mesh at $t = 100$ s and therefore no results are shown. Even with the coarse mesh the free surface model manages to obtain the correct interface position. With the enriched pressure two–phase model an important mass loss can be observed when using the coarse mesh.

The flow pattern is compared in Fig. 3.4. Again, using the finest mesh there is not much difference in the calculated fluid velocities between free surface model and the enriched pressure two–phase model. The velocities drawn in the empty region in the free surface case are simply the velocities used to transport the level set function. With the normal two–phase model spurious velocities that distort the whole flow field and ruin mass conservation can be observed close to the interface. The free surface model manages to obtain the correct flow field both with the medium and coarse meshes. With the enriched pressure two–phase model spurious velocities are obtained close to the interface with the medium mesh. They increase in the coarse mesh. For the normal two–phase model no results are shown for the medium and coarse meshes because, as we have already mentioned, the fluid has disappeared at $t = 100$ s.

The results obtained with the Eulerian simplified free surface model have not been

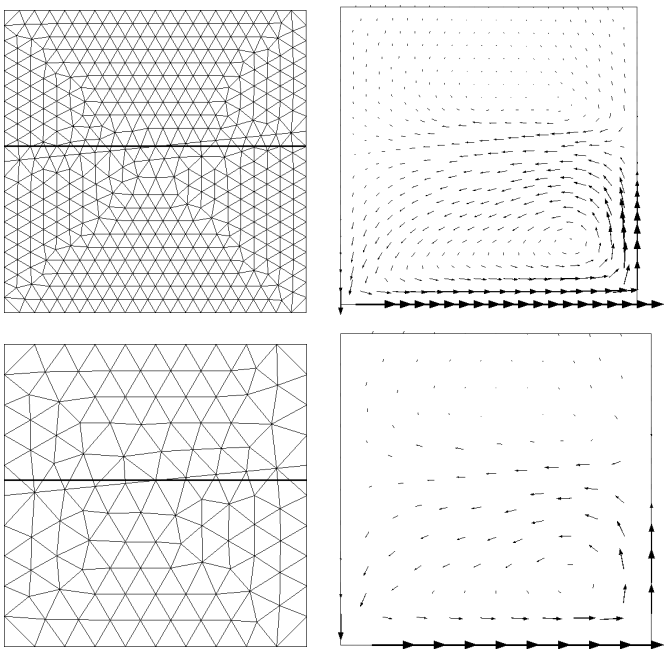


Figure 3.5: Flow field and interface position at $t = 100$ s using the Eulerian free surface model

shown up to now because they are very similar to those obtained with the FM-ALE free surface model. They are presented in Figure 3.5 for the medium and coarse meshes so that they can be compared with the results shown previously (Figures 3.3 and 3.4) for the FM-ALE model. In the rest of the examples presented in this Chapter both free surface models have always given nearly the same results and therefore only the results obtained with the FM-ALE model will be shown.

3.4.2 3D vertical channel

All the details (geometry, mesh, material properties, etc.) of the second example are identical to those presented in Chapter 2 and therefore they are not repeated.

In Figure 3.6 we show the shape of the interface and the velocity field obtained with the free surface FM-ALE model proposed in this Chapter. For the results with the enriched pressure two phase model see Figure 2.6. Both models provide the correct solution; the interface remains flat and its displacement corresponds to the amount of

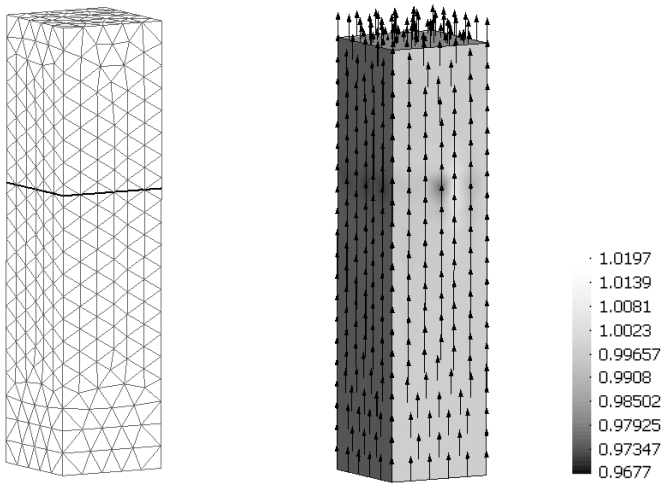


Figure 3.6: Interface shape and vertical velocity band plot together with velocity vectors using the FM-ALE free surface model

injected fluid. When the normal two phase flow model is used (see Fig. 2.8) the results are very poor. The velocity shows important oscillations close to the interface and there is an important mass loss. The mass loss is so important that the free surface remains nearly at its initial height. The source of the errors in the normal two phase flow model is the impossibility of the shape functions to capture the discontinuous pressure gradient that exists at the interface. One way to solve the problem is to enrich the pressure shape functions so as to represent the discontinuous pressure gradient more accurately. The other way to solve the problem is to use a free surface model and thus avoid the existence of a discontinuous pressure gradient by modeling only one fluid. The slight errors that remain in the velocity field with the two successful models can be attributed to the fact that the numerical resolution of the level set is not exact and therefore the deviations in the shape of the interface give rise to small variations in the velocity.

3.4.3 Sloshing problem

The third example coincides exactly with the one presented in Chapter 2 and the details are not repeated. The mesh and initial interface position have been given in Chapter 2.

In Fig. 3.7 we show the position of the interface after 11 seconds, using the free surface FM-ALE model. This results have to be compared with those presented in Fig. 2.11. Figure 3.8 shows the computed time history of $\eta\left(\frac{b}{2}, t\right)$ for the three cases together with the results presented by Ramaswamy et al. [100]. Using the normal two phase model there is a significant mass loss. The results obtained with the free surface and enriched pressure two phase models agree closely with the ones reported by Ramaswamy et al. [100]

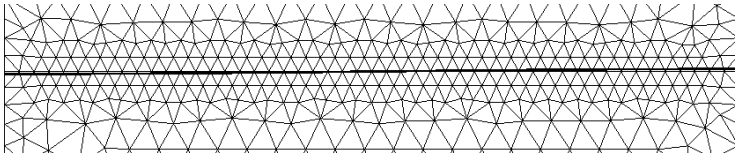


Figure 3.7: Interface position at $t = 11$s for the sloshing problem

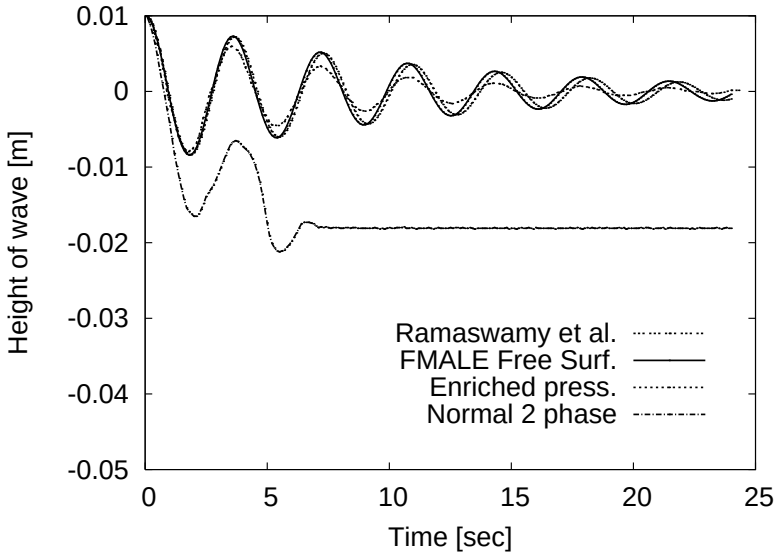


Figure 3.8: Time histories of surface elevation amplitude for the sloshing problem

3.5 Two computationally demanding examples

In the previous section simple numerical examples that show the benefits of the proposed method have been presented. In this Section two additional examples not included in [38]

are presented. The objective is to show that the methods works satisfactorily for complex problems and give some idea of the size of the problems that can be solved.

In Sections 3.2 and 3.3 we have explained how to extend the velocities to the empty region. We proposed to solve the Stokes or Navier Stokes equations in the empty region and mentioned that other options could also be used. In order to reduce the computational cost a cheaper velocity extrapolation is used in these examples. First the nodes in the empty region are classified into levels. The first level corresponds to the empty nodes of cut elements. Then, starting by $n = 1$, level $n+1$ is formed by the empty nodes connected to the level n until all of the nodes in the empty region have been classified. Once this has been done the velocities of each of the empty nodes with level greater than one, can be calculated as the average of velocities of the nodes in the lower level connected to it. The calculation is ordered so that all the nodes in the lower level are calculated before stepping to the next level. On the boundaries the velocities are corrected so that they satisfy Dirichlet boundary conditions. Despite this extension velocity is not divergence free we have found that it does not introduce any significant difference in the resulting flow in the fluid compared to solving the flow equations in the empty region and the computational cost is much lower. It is also used in Chapter 5 where industrial mould filling examples are presented.

3.5.1 3D dam-break wave interacting with a circular cylinder

The first problem is a 3D dam-break wave interacting with a circular cylinder borrowed from [77]. The domain is 20 m long and 5 m wide. In [77] a constant 10 m height has been used but we have preferred to increase the height at the right end of the domain because we have observed that otherwise the water would reach the upper surface and extend through it. The initial volume is 4 m long, 5 m wide and 7 m high. The circular cylinder, which has a radius $r = 1\ m$ and height $h = 5\ m$, is placed in the middle of the tank. An unstructured triangular mesh with 1968844 elements and 348963 nodes is used.

Water properties, $\rho = 1000.0$ and $\mu = 1.0\times10^{-3}$ (SI units), are used in the simulation.

The Reynolds number based on a typical velocity (10 m/s) and the diameter of the cylinder is $Re = 2.0 \times 10^7$. Despite the flow is turbulent, as the Reynolds number indicates, we have been able to run this example without using any turbulence model [49, 63, 95]. The viscosity introduced by the stabilization method seems to be enough to make the solution of the Navier Stokes equations possible. The walls of the domain are supposed frictionless. A total of 34 seconds have been run with a 0.01 time step.

In Figure 3.9.the evolution of the interface is shown. The results are similar to the ones obtained in [77] but not identical. This is not surprising due to complexity of the flow and the fact that no turbulence model has been used.

Split OSS stabilization has been used for the Navier Stokes equations. For the convective nonlinearity Picard iteration is used. The tolerance is set to one percent variation in the L2 norm of the velocity and a maximum of 10 iterations are allowed. Typically only one or two iterations are needed. For the solution of the monolithic system a preconditioned GMRES iterative solver [102] is used. The stopping criteria for the solver is that the residual is smaller than 10^{-8} times the right hand side. It usually converges in approximately 30 iterations. An ILUT preconditioner with threshold 0.001 and filling 25 is used [102]. For the Level Set equation the convergence of the GMRES solver is very easy even without preconditioner.

The total CPU time for the simulation has been 530932 seconds. The resolution of the Navier Stokes equations takes most of the time, with 232416 seconds for the matrix assembly and 225672 seconds for the linear solver. The runs were performed on a PC with AMD Athlon(tm) 64 X2 Dual Core Processor 4400+ running at 2.2 GHz with 3 Gbyte of RAM using the Intel Fortran compiler under Ubuntu.

3.5.2 3D Green water problem

The second example has been experimentally studied by the Maritime Research Institute Netherlands (MARIN) to evaluate the effects of green water flow over the deck of ships. As in the previous example it is a 3D dam-break wave but in this case it hits a box with

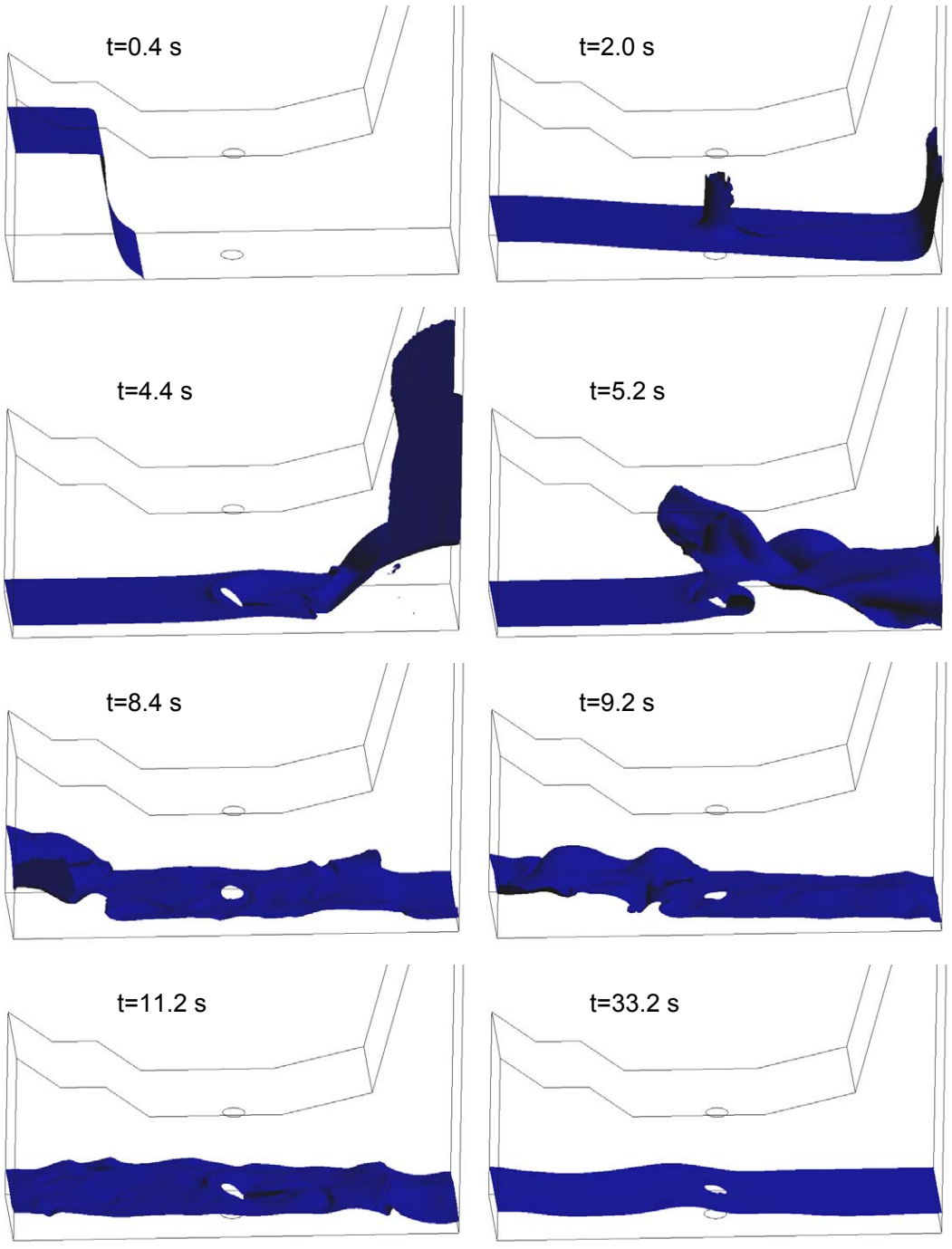


Figure 3.9: Free surface evolution

dimensions $0.161 \times 0.403 \times 0.161\ m$. The tank has an open roof of dimensions $3.22 \times 1 \times 1\ m$ and the initial water volume is $0.55\ m$ high and $1.228\ m$ long. The smaller dimensions make this problem somehow simpler than the previous one but the advantage is that experimental data is available in [35]. The problem has also been analyzed numerically in [41, 47, 69].

An unstructured triangular mesh with 1166780 elements and 206153 nodes is used. The numerical strategy is the same as in the previous example. A total of 6 seconds have been run with a 0.0025 time step. The total CPU time for the simulation has been 314644 seconds. The resolution of the Navier Stokes equations takes most of the time, with 116755 seconds for the matrix assembly and 112844 seconds for the linear solver.

The pressures at four of the gauges whose position is described in [35] are compared with our numerical results in Figure 3.10. P1 and P3 are located at the face that receives the wave impact, while P6 and P8 are found at the top of the box. The agreement between the numerical results and the experimental ones is very satisfactory. Some small delay ($0.4\ s$) can be observed for the moment the return wave hits the box again at about $5.0\ s$. The reason for this delay should be explored further.

In Figure 3.11.the evolution of the interface is shown. The agreement with the photographs presented in [35] is very satisfactory. It is important to point out that a smooth interface is obtained in the region far from the wave ($t = 2.3\ s$). This good behavior can be attributed to the correct treatment of boundary conditions at the interface.

3.6 Conclusions

In this Chapter we have introduced a formulation for free surface flows on fixed meshes. The key difference with typical Eulerian formulations for free surface flows is that we do not solve the Navier–Stokes equations on the whole mesh. Taking advantage of the fact that we have a boundary condition at the free surface we can model the flow only in the fluid region. By doing so we are able to avoid the difficulty of modeling a discontinuous

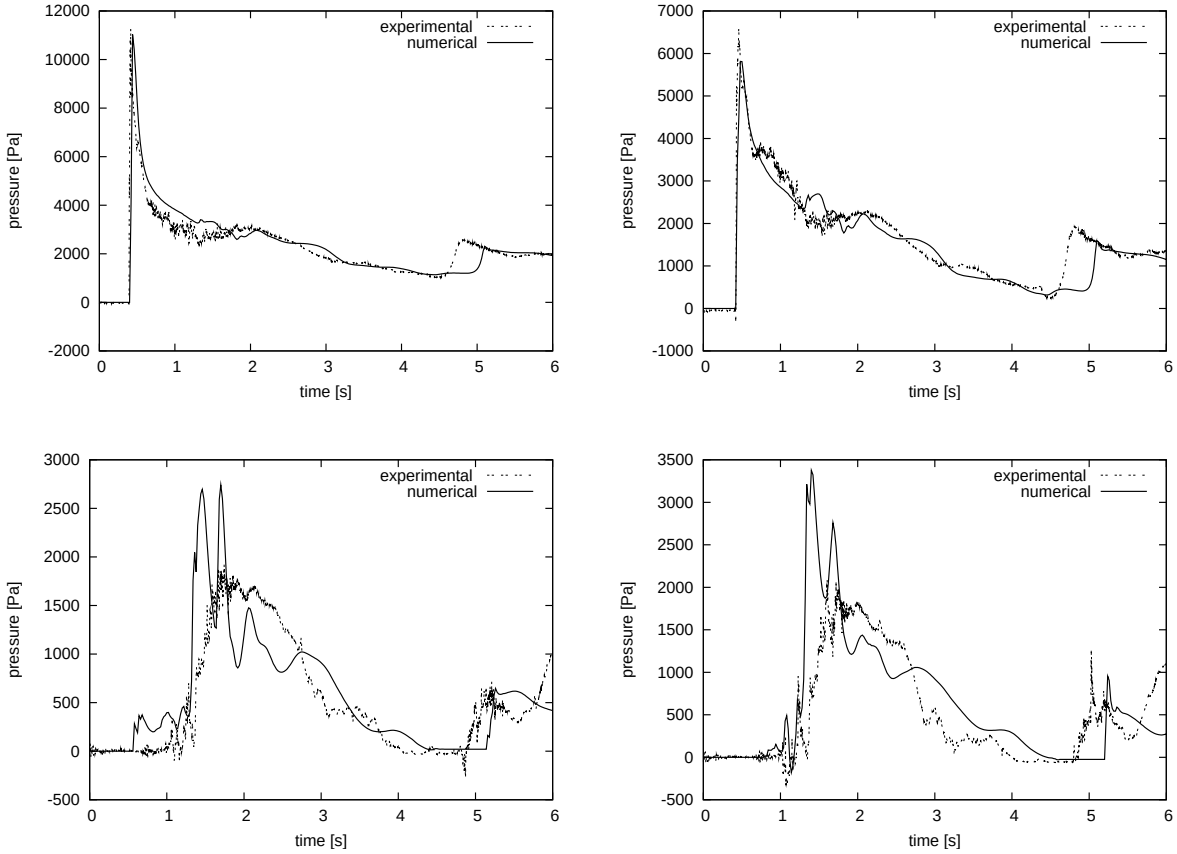


Figure 3.10: Pressure at points P1, P3, P6 and P8

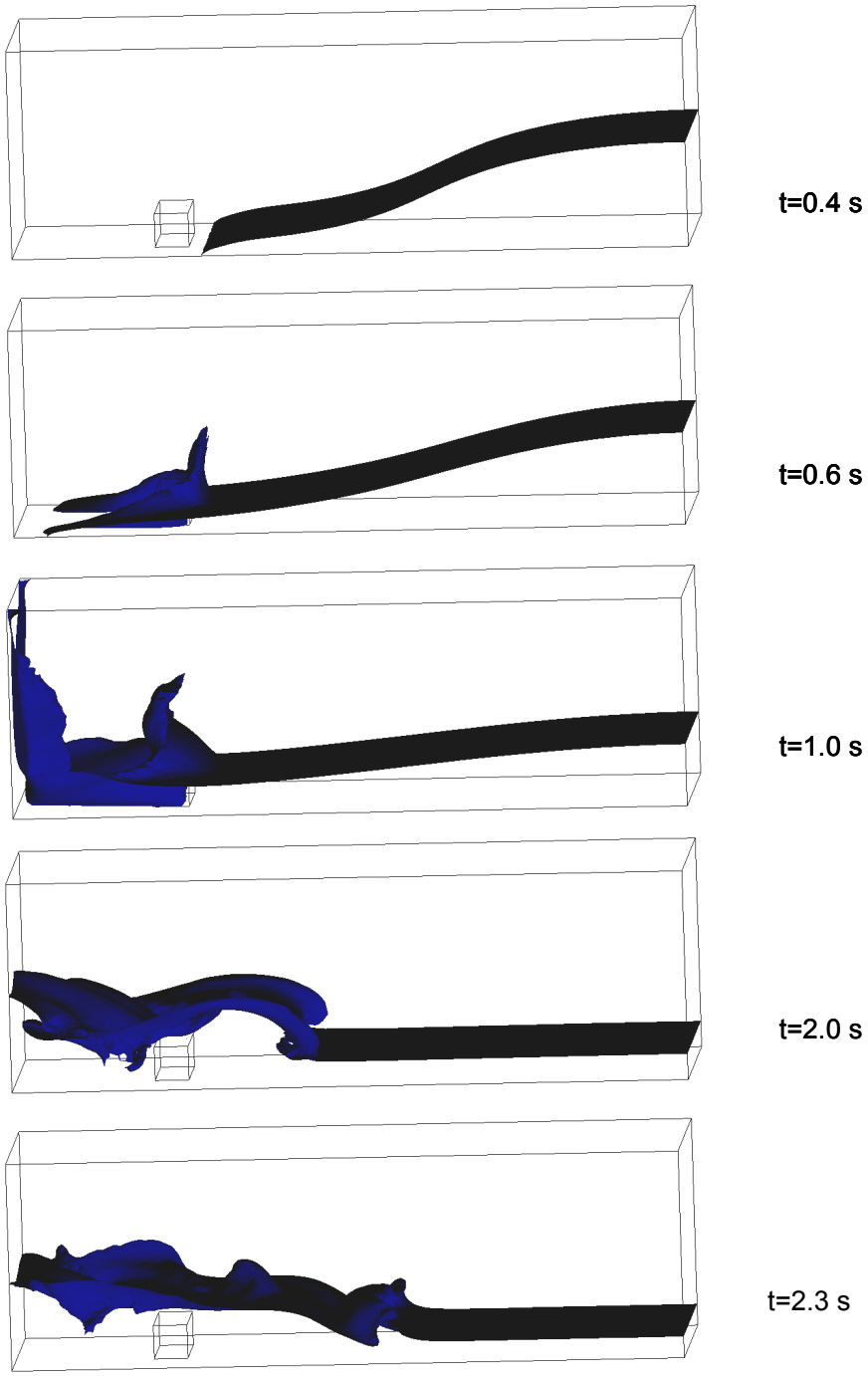


Figure 3.11: Free surface evolution for the 3D Green water problem

pressure gradient in the cut elements. The use of enhanced integration allows us to impose the Neumann boundary condition at a surface that does not coincide with element faces quite simply. On the other hand the model differs from ALE or Lagrangian simulations on the fact that we use a fixed mesh. Two versions of the model have been proposed, one that uses an ALE formulation on fixed meshes (FM-ALE) and the other one that is more closely related to Eulerian formulations. The second formulation is a simplification or approximation of the first one. Nevertheless, no significant difference has been observed between the numerical results obtained with the two versions.

The FM-ALE version calculates the velocities in the empty region only to transport the level set function. The other version also uses the velocities in the empty region to model the transient term in nodes that go from the empty region to the filled one in a way that mimics what happens in a two phase flow with negligible properties in the second fluid. The model takes into account the fact that in a free surface flow the velocities in the fluid region influence the velocities in the empty region but not the other way around.

The numerical results have shown that the formulation can provide very satisfactory results for low Froude number flows where the typical two phase flow model fails. Compared with the enriched pressure two phase model presented in Chapter 2 the free surface model has produced equal or better results in all the analyzed cases.

The free surface model is simpler than the enriched pressure two phase model because no enrichment is needed. On the other hand, since the flow equations are solved in the liquid and the empty region separately, the computational cost is reduced.

The examples shown in this paper demonstrate that the free surface model can introduce significant improvements compared with a typical two phase flow finite element model. It allows to avoid spurious velocities and enhances mass conservation.

Chapter 4

Pressure Segregation Methods

In this Chapter we will explore pressure segregation methods that should allow us reduce the computational cost of solving the Navier-Stokes equations. Since their appearance in the late 1960's with the works of Chorin [20] and Teman [107] have enjoyed widespread popularity. The key for such success is that they allow to uncouple the velocity and pressure unknowns, leading not only to smaller, but also, better conditioned subproblems. The idea is to continue the work presented by Santiago Badia in a recent thesis [4] concentrating mainly on the implementation and numerical testing of the algorithms presented therein, specially for the interface flows we are interested in. The objective of this chapter is to gain some practical experience with pressure segregation methods, test our implementations and select which method we should use for our mould filling problems.

Following [4] pressure segregation methods can be classified into pressure correction methods and velocity correction methods. The former are the most well known and include the Chorin-Teman projection method and the Van Kan method [117]. The latter are more recent [52, 67]. The velocity correction approach we will use has been developed in [4] . Both versions will be tested. A complete review on pressure segregation methods can be found in [50]. They are usually also called projection methods or fractional step methods.

The most typical approach is to first uncouple the velocity and pressure at the space

continuous level and then discretize the problem. The approach we will use in this work is to introduce the splitting at the purely algebraic level, once the discretization has been performed. Such approach is advocated in [93,96,113]. The main difference between both approaches is the way in which the boundary conditions are approximated. In any case, when using the discrete approach a further approximation is usually introduced (specially when dealing with continuous pressure interpolations) that makes both approaches very similar.

Predictor corrector (see [11] and references therein) versions of both velocity correction and pressure correction methods will also be presented. These methods obtain the splitting in a very similar way to the previous methods, but an iterative procedure is introduced that, when converged, leads to the same solution as the monolithic system. In fact, the non predictor corrector versions can be seen as the first iteration of the predictor corrector versions.

Finally we will reinterpret pressure correction schemes as an iterative procedure for solving the Pressure Schur Complement as suggested by Turek in [113, 115]. This new perspective is interesting because depending on the preconditioner chosen other well known methods (such as the SIMPLE scheme) can also be described. On the other hand, using such approach, the rotational version of the pressure correction schemes, introduced in [111] and highly favored in [50], can be seen as the use of a different preconditioner from the one used to obtain the standard version. In fact, it is the preconditioner introduced in [17] and recommended in [115]. In [4] the use of the rotational version has not been tested because it is not considered necessary when the splitting is done at the purely algebraic level (see remark 3.1). In [115], despite a discrete splitting is used, the rotational version is preferred and therefore we believe that it could be interesting to test it.

4.1 Pressure correction methods

4.1.1 Fractional Step (non Predictor Corrector) schemes

As we have already mentioned, the splitting will be introduced at the pure algebraic level, that is, starting from the monolithic discretized problem written in matrix form (1.10). In order to simplify the presentation, only homogeneous Dirichlet boundary conditions will be taken into account and the stabilization terms will not be included. Using a BDF1 time discretization, the problem then reads:

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}+\mathsf{G}\mathsf{P}^{n+1}=\mathsf{F}^{n+1},$$

$$\mathsf{D}\mathsf{U}^{n+1}=\mathsf{0}.$$

The spliting can now be introduced, giving rise to an *exactly equivalent* problem

$$\frac{1}{\delta t}\mathsf{M}\left(\widetilde{\mathsf{U}}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}+\gamma\mathsf{G}\mathsf{P}^{n}=\mathsf{F}^{n+1}, \tag{4.1}$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\widetilde{\mathsf{U}}^{n+1}\right)+\mathsf{G}\left(\mathsf{P}^{n+1}-\gamma\mathsf{P}^{n}\right)=\mathsf{0}, \tag{4.2}$$

$$\mathsf{D}\mathsf{U}^{n+1}=\mathsf{0}, \tag{4.3}$$

where $\widetilde{\mathsf{U}}^{n+1}$ is an auxiliary variable and γ is a numerical parameter, whose values of interest are 0 and 1. At this point the first and essential approximation is introduced

$$\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}\approx\mathsf{K}\left(\widetilde{\mathsf{U}}^{n+1}\right)\widetilde{\mathsf{U}}^{n+1}. \tag{4.4}$$

Expressing U^{n+1} in terms of $\widetilde{\mathsf{U}}^{n+1}$ using 4.2 and inserting the result in 4.3, the set of equations to be solved is

$$\frac{1}{\delta t}\mathsf{M}\left(\widetilde{\mathsf{U}}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\widetilde{\mathsf{U}}^{n+1}\right)\widetilde{\mathsf{U}}^{n+1}+\gamma\mathsf{G}\mathsf{P}^{n}=\mathsf{F}^{n+1}, \tag{4.5}$$

$$\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\left(\mathsf{P}^{n+1}-\gamma\mathsf{P}^{n}\right)=\mathsf{D}\widetilde{\mathsf{U}}^{n+1}, \tag{4.6}$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\widetilde{\mathsf{U}}^{n+1}\right)+\mathsf{G}\left(\mathsf{P}^{n+1}-\gamma\mathsf{P}^{n}\right)=\mathsf{0}. \tag{4.7}$$

which has been ordered according to the sequence of solution, for $\widetilde{\mathsf{U}}^{n+1}$, P^{n+1} and U^{n+1}. The uncoupling of the variables has been made possible thanks to approximation 4.4.

Depending mainly on the type of finite element discretization used, a second approximation is typically introduced. When continuous pressure interpolations are used, it is much cheaper to approximate

$$\mathsf{D}\mathsf{M}^{-1}\mathsf{G} \approx \mathsf{L}, \qquad \text{with components} \quad \mathsf{L}^{ab} = -\left(\nabla N^a, \frac{1}{\rho}\nabla N^b\right) \tag{4.8}$$

where L is the standard approximation to the Laplacian operator divided by the density when a constant density is used. When this approximation is used the system turns out to be very similar to the one that would be obtained if the splitting had been introduced prior to the discretization. If $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ needs to be explicitly build, as in [114], a diagonal M matrix has to be used. It can be obtained either by lumping the standard one or by using nodal integration. The extension of projection methods to variable density flows is quite recent [10, 51]

When the second approximation ($\mathsf{D}\mathsf{M}^{-1}\mathsf{G} \approx \mathsf{L}$) is used and $\gamma = 0$ the original scheme proposed by Chorin [20] and Teman [107] is recovered. It is usually referred to as non-incremental scheme. When $\gamma = 1$ is used the incremental version, introduced by Van Kan [117], is obtained.

If a BDF2 time discretization had been used, following the same steps we would have arrived to

$$\frac{1}{2\delta t}\mathsf{M}\left(3\widetilde{\mathsf{U}}^{n+1} - 4\mathsf{U}^n + \mathsf{U}^{n-1}\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^n = \mathsf{F}^{n+1},$$

$$\frac{2}{3}\delta t \mathsf{D}\mathsf{M}^{-1}\mathsf{G}\left(\delta\mathsf{P}^{n+1}\right) = \mathsf{D}\widetilde{\mathsf{U}}^{n+1},$$

$$\frac{1}{2\delta t}\mathsf{M}\left(3\mathsf{U}^{n+1} - 3\widetilde{\mathsf{U}}^{n+1}\right) + \mathsf{G}\left(\delta\mathsf{P}^{n+1}\right) = \mathsf{0}.$$

where $\delta\mathsf{P}^{n+1} = \mathsf{P}^{n+1} - \mathsf{P}^n$, using the notation introduced in Chapter 1.

A remarkable fact about the previous schemes is that, despite the pressure gradient is treated explicitly in the equation for $\widetilde{\mathsf{U}}^{n+1}$, they turn out to be stable in time. For a complete review on stability results see [5, 52].

The name pressure correction that we shall use in this work originates from the fact in the incremental version a first order extrapolation of the pressure $\widetilde{\mathsf{P}}^{n+1} = \mathsf{P}^n$ is used

to obtain $\widetilde{\mathsf{U}}^{n+1}$ and in the second step a correction $\delta\mathsf{P}^{n+1}$ is obtained. Also in the non-incremental case an extrapolation of the pressure is used, in this case of order zero, $\widetilde{\mathsf{P}}^{n+1} = \mathsf{0}$. The error introduced by the splitting is one order higher that the error of the extrapolation used for $\widetilde{\mathsf{P}}^{n+1}$. If the error $\left\|\mathsf{P}^{n+1} - \widetilde{\mathsf{P}}^{n+1}\right\|$ in any norm $\|.\|$ is of order p, then from (4.7) we can see that $O\left(\left\|\mathsf{U}^{n+1} - \widetilde{\mathsf{U}}^{n+1}\right\|\right) = \delta t \; O\left(\left\|\mathsf{P}^{n+1} - \widetilde{\mathsf{P}}^{n+1}\right\|\right) = p + 1$. The non-incremental scheme has a splitting error of order 1 and the incremental one of order 2. In this work the incremental version will always be used and the non-incremental scheme will no longer be presented. The temporal order of the method is given by the minimum between the order of the error introduced by the discretization of the temporal derivative and the order of the error introduced by the splitting. Even if a first order time discretization (BDF1) is used, the use of the incremental version will be advantageous when the splitting error is bigger than the error introduced by the discretization of the temporal derivative. The idea of using higher order extrapolations for the pressure seems tempting. The order of the spliting error would be reduced and the solution would rapidly tend to the monolithic solution. Unfortunately numerical experience [4, 52] shows that pressure extrapolations of order higher than one typically lead to unstable solutions.

In the introduction we have mentioned that the schemes obtained at the discrete [4,93,96,113] and continuous [20,50,107] level are very similar when the discrete Laplacian is approximated by the continuous one. The difference is the Dirichlet boundary condition applied on the end of step velocity U. When the splitting is introduced at the continuous level only the normal component is prescribed. Instead when the discrete approach is used all components are prescribed.

4.1.2 Predictor Corrector scheme

The predictor corrector method we will describe has been proposed in [33, 105]. Without taking into account the stabilization terms and using a BDF1 discretization it reads

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1,i+1} - \mathsf{U}^{n}\right) + \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1} + \mathsf{G}\mathsf{P}^{n+1,i} = \mathsf{F}^{n+1}, \tag{4.9}$$

$$\delta t\mathsf{L}\left(\mathsf{P}^{n+1,i+1} - \mathsf{P}^{n+1,i}\right) = \mathsf{D}\mathsf{U}^{n+1,i+1}, \tag{4.10}$$

where the superscript i indicates the iteration number. In [4,33,105] it is supposed that the uncoupling is dealt with in the same iterative loop as the one used for the linearization of the convective term. In our implementation we have used nested loops for the uncoupling and the linearization of the convective term. The uncoupling loop has been set as the outer loop. In this way, when only one linearization iteration is permitted per uncoupling iteration, the usual predictor corrector method with a single loop is recovered. On the other hand when only one uncoupling iteration is permitted the fractional step version is recovered.

It is obvious that when the method converges ($\mathsf{P}^{n+1,i+1} = \mathsf{P}^{n+1,i}$) the solution of the original monolithic scheme (1.10) is recovered. $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ could have been used instead of L. How this would affect convergence could be interesting to analyze. The inclusion of the term $\delta t \mathsf{L}\left(\mathsf{P}^{n+1,i+1} - \mathsf{P}^{n+1,i}\right)$ is motivated by what happens in the fractional step (non predictor corrector) scheme.

If instead of using a first order scheme, a BDF2 time discretization had been used, the iterative scheme would read [26]

$$\frac{1}{2\delta t}\mathsf{M}\left(\mathsf{U}^{n+1,i+1} - 4\mathsf{U}^{n} + \mathsf{U}^{n-1}\right) + \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1} + \mathsf{G}\mathsf{P}^{n+1,i} = \mathsf{F}^{n+1},$$

$$\delta t \frac{2}{3}\mathsf{L}\left(\mathsf{P}^{n+1,i+1} - \mathsf{P}^{n+1,i}\right) = \mathsf{D}\mathsf{U}^{n+1,i+1}.$$

Contrary to what happens in the fractional step version, for $\mathsf{P}^{n+1,0}$ and $\mathsf{U}^{n+1,0}$ extrapolations of order higher than one can be used without compromising the stability because at each time step the method converges to the monolithic solution. In [26] the use of a second order extrapolation reduces the number iterations needed to converge to the monolithic solution compared to a first order extrapolation in some numerical examples.

In [33] the predictor corrector version is initially written with both $\widetilde{\mathsf{U}}$ and U, but as in the converged case $\widetilde{\mathsf{U}} = \mathsf{U}$, finally the scheme is written with only one velocity as in (4.9,4.10). In practice one usually does not converge up the absolute zero but only up to some finite tolerance or even some fixed number of iterations. In such case, we believe that one should work with both velocities ($\widetilde{\mathsf{U}}$ and U). The split predictor corrector scheme,

using a BDF1 time discretization, would then read:

$$\frac{1}{\delta t}\mathsf{M}\left(\widetilde{\mathsf{U}}^{n+1,i+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\widetilde{\mathsf{U}}^{n+1,i}\right)\widetilde{\mathsf{U}}^{n+1,i+1}+\mathsf{G}\mathsf{P}^{n+1,i}=\mathsf{F}^{n+1}, \tag{4.11}$$

$$\delta t\mathsf{L}\left(\mathsf{P}^{n+1,i+1}-\mathsf{P}^{n+1,i}\right)=\mathsf{D}\widetilde{\mathsf{U}}^{n+1,i+1}, \tag{4.12}$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\widetilde{\mathsf{U}}^{n+1,i+1}\right)+\mathsf{G}\left(\mathsf{P}^{n+1,i+1}-\mathsf{P}^{n+1,i}\right)=0. \tag{4.13}$$

Equation 4.13 is solved at the end of the iterative process, that is when (4.11) and (4.12) have converged to the user prescribed tolerance.

The OSS stabilized scheme

So as to conclude this section we will present the Split OSS stabilized version of the pressure correction predictor corrector scheme using a BDF1 time discretization and the matrices introduced in Chapter 2. Using the notation introduced in Chapter 1 the matrix version of the problem reads as follows:

$$\begin{gathered}\frac{1}{\delta t}\mathsf{M}\left(\widetilde{\mathsf{U}}^{n+1,i+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\widetilde{\mathsf{U}}^{n+1,i}\right)\widetilde{\mathsf{U}}^{n+1,i+1}+\mathsf{G}\mathsf{P}^{n+1,i}\\ +\mathsf{S}_u\left(\tau_1^{n+1,i};\widetilde{\mathsf{U}}^{n+1,i}\right)\widetilde{\mathsf{U}}^{n+1,i+1}-\mathsf{S}_y\left(\tau_1^{n+1,i};\widetilde{\mathsf{U}}^{n+1,i}\right)\mathsf{Y}^{n+1,i}\\ +\mathsf{S}_d\left(\tau_2^{n+1,i}\right)\widetilde{\mathsf{U}}^{n+1,i+1}-\mathsf{S}_w\left(\tau_2^{n+1,i}\right)\mathsf{W}^{n+1,i}=\mathsf{F}^{n+1},\end{gathered}$$

$$\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\left(\mathsf{P}^{n+1,i+1}-\mathsf{P}^{n+1,i}\right)-\mathsf{S}_p\left(\tau_1^{n+1,i+1}\right)\mathsf{P}^{n+1,i+1}+\mathsf{S}_z\left(\tau_1^{n+1,i+1}\right)\mathsf{Z}^{n+1}-\mathsf{D}\widetilde{\mathsf{U}}^{n+1,i+1}=0,$$

$$\mathsf{M}_\pi\mathsf{Y}^{n+1,i+1}-\mathsf{C}\left(\widetilde{\mathsf{U}}^{n+1,i+1}\right)\widetilde{\mathsf{U}}^{n+1,i+1}=0,$$

$$\mathsf{M}_\pi\mathsf{Z}^{n+1,i+1}-\mathsf{G}_\pi\mathsf{P}^{n+1,i+1}=0,$$

$$\mathsf{M}_\pi\mathsf{W}^{n+1,i+1}-\mathsf{D}\widetilde{\mathsf{U}}^{n+1,i+1}=0,$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\widetilde{\mathsf{U}}^{n+1,i+1}\right)+\mathsf{G}\left(\mathsf{P}^{n+1,i+1}-\mathsf{P}^{n+1,i}\right)=0.$$

We have included the terms corresponding to $\tau_2 \neq 0$ that have been neglected in [4,33,105]. These terms help to enforce incompressibility. In two fluid flows, where it is important to conserve the mass of each fluid, the influence of such terms is something that seems interesting to explore numerically.

If the difference between U and $\widetilde{\mathsf{U}}$ is neglected, as has been done in the previous publications [4, 33, 105], it has been observed [33, 105] that the previous equations can be seen as an iterative procedure for solving the *monolithic* problem freezing the pressure gradient in the momentum equation. The term $\delta t \mathsf{L}\left(\mathsf{P}^{n+1,i+1} - \mathsf{P}^{n+1,i}\right)$ has been motivated by what happens in the fractional step case. If it had been omitted the same iterative procedure could have been used thanks to the inclusion of matrix S_p. In [33, 105] both options have been tested and it has been pointed out that without the inclusion of the Laplacian convergence turns out to be much harder.

4.2 The Pressure Schur Complement approach

Following Turek [113,115], in this section we will present the Pressure Schur Complement approach that allows us to obtain another interpretation of pressure correction methods. An interesting feature of this approach is that is allows to describe not only pressure corrections methods but also other well known solution schemes, such as SIMPLE or Uzawa iterations. Related approaches can also be found in [5].

In order to present the method we will start from the matrix version of Navier Stokes equations obtained after discretization both in space and time (BDF1) using the notation introduced in Chapter 1. For simplicity the stabilization terms will be omitted. The problem is then to find U^{n+1} and P^{n+1} given U^n, F^{n+1} and δt such that

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1} - \mathsf{U}^n\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} = \mathsf{F}^{n+1},$$

$$\mathsf{D}\mathsf{U}^{n+1} = \mathsf{0}.$$

The previous equations can be rewritten as

$$\mathsf{M}\mathsf{U}^{n+1} + \delta t \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \delta t \mathsf{G}\mathsf{P}^{n+1} = \delta t \mathsf{F}^{n+1} + \mathsf{M}\mathsf{U}^n,$$

$$\mathsf{D}\mathsf{U}^{n+1} = \mathsf{0}.$$

where in order to adapt the presentation to the one used in [115] S and F^* are defined:

$$\mathsf{S} = \mathsf{M} + \delta t \mathsf{K}\left(\mathsf{U}^{n+1}\right)$$

$$\mathsf{F}^{*}=\delta t\mathsf{F}^{n+1}+\mathsf{M}\mathsf{U}^{n}$$

Then the (nonlinear) algebraic problem to be solved is: given U^n, F^* and δt, solve for $\mathsf{U}=\mathsf{U}^{n+1}$ and $\mathsf{P}=\mathsf{P}^{n+1}$

$$\mathsf{S}\mathsf{U}+\delta t\mathsf{G}\mathsf{P}=\mathsf{F}^{*}, \tag{4.14}$$

$$\mathsf{D}\mathsf{U}=\mathsf{0}. \tag{4.15}$$

S and F^* may vary depending on the time discretization or on the linearization used for the convective terms but the system to be solved is always of the form (4.14,4.15).

Assuming S^{-1} exists, problem (4.14,4.15) is equivalent to the following scalar pressure Schur complement formulation

$$-\mathsf{D}\mathsf{S}^{-1}\mathsf{G}\mathsf{P}=-\frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1}\mathsf{F}^{*}. \tag{4.16}$$

Once the pressure P is known, the corresponding velocity vector U satisfies

$$\mathsf{U}=\mathsf{S}^{-1}\left(\mathsf{F}^{*}-\delta t\mathsf{G}\mathsf{P}\right).$$

The idea is to present the problem as a scalar problem for P so that the knowledge about efficient iterative schemes for such problems can be applied. One of the possibilities is to perform a *preconditioned Richardson* iteration, where C^{-1} is an appropriate preconditioner for the pressure Schur complement $-\mathsf{D}\mathsf{S}^{-1}\mathsf{G}$. The basic iteration for the pressure Schur complement equation is then: given P^i obtain P^{i+1}

$$\mathsf{P}^{i+1}=\mathsf{P}^{i}-\mathsf{C}^{-1}\left(-\mathsf{D}\mathsf{S}^{-1}\mathsf{G}\ \mathsf{P}^{i}+\frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1}\mathsf{F}^{*}\right). \tag{4.17}$$

In [115] it is proposed that

$$\mathsf{C}^{-1}=\alpha_R\mathsf{T}^{-1}+\alpha_D\mathsf{M}_P^{-1}, \tag{4.18}$$

where $\mathsf{T}:=-\mathsf{D}\mathsf{M}_L^{-1}\mathsf{G}$, M_L and M_P are the diagonal (lumped) mass matrices corresponding to the velocity and pressure respectively and α_R, α_D are damping parameters. We then

have

$$\begin{aligned} \mathsf{P}^{i+1} &= \mathsf{P}^i - \left[\alpha_R \mathsf{T}^{-1} + \alpha_D \mathsf{M}_P^{-1}\right] \left(-\mathsf{D}\mathsf{S}^{-1}\mathsf{G}\ \mathsf{P}^i + \frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1}\mathsf{F}^*\right) \\ &= \mathsf{P}^i + \left[\alpha_R \mathsf{T}^{-1} + \alpha_D \mathsf{M}_P^{-1}\right] \left(-\frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1}\mathsf{F}^* + \frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1}\delta t\mathsf{G}\ \mathsf{P}^i\right) \\ &= \mathsf{P}^i + \left[\alpha_R \mathsf{T}^{-1} + \alpha_D \mathsf{M}_P^{-1}\right] \left(-\frac{1}{\delta t}\mathsf{D}\underbrace{\mathsf{S}^{-1}\left(\mathsf{F}^* - \delta t\mathsf{G}\ \mathsf{P}^i\right)}_{\widetilde{\mathsf{U}}^{i+1}}\right) \end{aligned} \tag{4.19}$$

Now the previous iterative procedure can be split into 4 substeps that are equivalent to those we have proposed for the predictor-corrector scheme in the previous section. Given P^i and F^* perform the following 4 substeps to obtain P^{i+1}:

1. Solve for $\widetilde{\mathsf{U}}^{i+1}$

$$\mathsf{S}\widetilde{\mathsf{U}}^{i+1} = \mathsf{F}^* - \delta t\mathsf{G}\ \mathsf{P}^i$$

2. Calculate a right hand side F_P for the preconditioning step

$$\mathsf{F}_P = -\frac{1}{\delta t}\mathsf{D}\widetilde{\mathsf{U}}^{i+1}$$

3. Solve an update-equation for the pressure

$$\mathsf{T}\mathsf{Q}^i = \mathsf{F}_P$$

4. Update the new pressure

$$\mathsf{P}^{i+1} = \mathsf{P}^i + \alpha_R\mathsf{Q}^i + \alpha_D\mathsf{M}_P^{-1}\mathsf{F}_P \tag{4.20}$$

Step 1 corresponds to the first step of our predictor corrector scheme (4.11). When $\alpha_R = 1$ and $\alpha_D = 0$ steps 2, 3 and 4 are equivalent to the second step of the predictor corrector scheme (4.12). The step that obtains U (4.13) has not been included because as in the predictor corrector case it is performed once the iterative procedure has converged. Using the notation we have just introduced it reads

$$\mathsf{U} = \widetilde{\mathsf{U}} - \delta t\mathsf{M}_L^{-1}\mathsf{G}\ \mathsf{Q}$$

where we have omitted the superindexes for $\widetilde{\mathsf{U}}$ and Q to indicate that they correspond to the converged solutions. If only one iteration is performed the fractional step scheme is recovered.

If $\alpha_R = 1$ and $\alpha_D = \mu\ \delta t$ the rotational version of the pressure correction schemes introduced in [111] and highly favored in [50] is obtained. The difference is that, as presented in [115], it is derived on the discrete level as an optimal preconditioner for the Stokes part of the Schur complement equation instead of as modifications to the differential operators at the continuous level. In [4] the rotational version is not considered necessary when the splitting is done at the purely algebraic level (see remark 3.1). Since in [115] splitting is introduced at the discrete level and the preconditioner with a non-zero α_D (corresponding to the rotational form) is preferred, we believe that it would be interesting to test it for our problems. Recently, in [5] it has been recognized that remark 3.1 in [4] was not correct. It is clarified that the error in the pressure close to Dirichlet boundaries is also present when the splitting is done at the purely algebraic and the discrete Laplacian is used.

An alternative interpretation for the error in the pressure close to Dirichlet boundaries can be found in [15]. Despite the solution is coincident with the rotational version no reference is made to [111]. Why the standard fractional step scheme introduces a spurious boundary condition close to Dirichlet boundaries and how the rotational form corrects this error is easier to see when the fractional step scheme is introduced at the continuous level than when it is introduced at the discrete level, as we do in this work. Therefore we refer the reader to [50, 111] where the continuous approach is used.

Rotational form for pressure stabilized schemes

When we tried to implement the rotational version of the fractional step scheme we found that it had not been applied to pressure stabilized elements. From (4.20) one can see that the pressure is obtained by adding two corrections to extrapolation of the pressure $\widetilde{\mathsf{P}}^{n+1}(= \mathsf{P}^n$ in the incremental case). The first correction is the one used both in the standard and rotational versions of the method and the second one only in the rotational

version. We can call them $\delta \mathsf{P}_1$ and $\delta \mathsf{P}_2$ respectively. How to obtain $\delta \mathsf{P}_1$ in the pressure stabilized case has already been described and is well known. In order to obtain $\delta \mathsf{P}_2$ a naive approach can be to proceed as in the non stabilized case, that is

$$\delta \mathsf{P}_2 = \alpha_D \mathsf{M}_P^{-1} \mathsf{F}_P = -\mu \mathsf{M}_P^{-1} \mathsf{D} \widetilde{\mathsf{U}}^{i+1}.$$

When we implemented this option pressure stability was lost.

In the non stabilized case, the approximation to the inverse of the pressure Schur complement (4.18) is built as the sum of two inverses. The first one approximates the inverse of the pressure Schur complement closely when the transient term is more important than the viscous and convective terms (it is related to $\delta \mathsf{P}_1$) and the second one when the viscous term is dominant (it is related to $\delta \mathsf{P}_2$). For the pressure stabilized case we now proceed as in the non stabilized case. Using Split OSS and only stabilizing the pressure to simplify the presentation, the algebraic problem to be solved is: given U^n, F^* and δt, solve for $\mathsf{U} = \mathsf{U}^{n+1}$ and $\mathsf{P} = \mathsf{P}^{n+1}$

$$\mathsf{S}\mathsf{U} + \delta t \mathsf{G}\mathsf{P} = \mathsf{F}^*,$$

$$\mathsf{D}\mathsf{U} + \mathsf{S}_p \mathsf{P} = \mathsf{S}_z \mathsf{Z}.$$

It leads to the following scalar pressure Schur complement formulation

$$\left(-\mathsf{D}\mathsf{S}^{-1}\mathsf{G} + \frac{1}{\delta t}\mathsf{S}_p \right) \mathsf{P} = \frac{1}{\delta t}\mathsf{S}_z \mathsf{Z} - \frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1} \left(\mathsf{F}^* \right).$$

As in the non stabilized case we can define the approximation to the pressure Schur complement as the sum of two inverses $\mathsf{C}^{-1} = \mathsf{C}_1^{-1} + \mathsf{C}_2^{-1}$. The first one should approximate the pressure Schur complement closely when the transient term dominates and the second one when the viscous term dominates. Similarly to what is done in the non stabilized case for the stabilized case we have

$$\mathsf{C}_1^{-1} = \left(-\mathsf{D}\mathsf{M}_L^{-1}\mathsf{G} + \frac{1}{\delta t}\mathsf{S}_p \right)^{-1}$$

and

$$\mathsf{C}_2^{-1} = \left(\frac{1}{\mu \delta t}\mathsf{M}_P + \frac{1}{\delta t}\mathsf{S}_p \right)^{-1}.$$

In the stabilized case C_2 is no longer diagonal and a system needs to be solved for $\delta\mathsf{P}_2$

$$\left(\frac{1}{\mu}\mathsf{M}_P + \mathsf{S}_p\right)\delta\mathsf{P}_2 = -\mathsf{D}\widetilde{\mathsf{U}}^{i+1} - \mathsf{S}_p\mathsf{P}^i + \mathsf{S}_z\mathsf{Z}.$$

Fortunately the matrix is well conditioned and the system is easy to solve.

We can now proceed as we have done in the non stabilized case to show that one iteration of the preconditioned Richardson iteration for the pressure Schur complement corresponds to the fractional step scheme (standard or rotational form). By doing this we can arrive to the fractional step pressure stabilized rotational scheme written in the usual form and show the error introduced by the splitting. We now rewrite the preconditioned Richardson iteration for the stabilized pressure Schur complement

$$\begin{aligned}
\mathsf{P}^{i+1} &= \mathsf{P}^i - \left[\mathsf{C}_1^{-1} + \mathsf{C}_2^{-1}\right]\left(-\mathsf{D}\mathsf{S}^{-1}\mathsf{G}\ \mathsf{P}^i + \frac{1}{\delta t}\mathsf{D}\mathsf{S}^{-1}\mathsf{F}^* + \frac{1}{\delta t}\mathsf{S}_p\mathsf{P}^i - \frac{1}{\delta t}\mathsf{S}_z\mathsf{Z}\right) \\
&= \mathsf{P}^i + \left[\mathsf{C}_1^{-1} + \mathsf{C}_2^{-1}\right]\left(-\frac{1}{\delta t}\mathsf{D}\underbrace{\mathsf{S}^{-1}\left(\mathsf{F}^* - \delta t\mathsf{G}\ \mathsf{P}^i\right)}_{\widetilde{\mathsf{U}}^{i+1}} - \frac{1}{\delta t}\mathsf{S}_p\mathsf{P}^i + \frac{1}{\delta t}\mathsf{S}_z\mathsf{Z}\right).
\end{aligned}$$

As in the non stabilized case we can now split the iterative procedure into 4 substeps

1. Solve for $\widetilde{\mathsf{U}}^{i+1}$

$$\mathsf{S}\widetilde{\mathsf{U}}^{i+1} = \mathsf{F}^* - \delta t\mathsf{G}\ \mathsf{P}^i$$

2. Calculate a right hand side F_P for the preconditioning step

$$\mathsf{F}_P = -\frac{1}{\delta t}\mathsf{D}\widetilde{\mathsf{U}}^{i+1} - \frac{1}{\delta t}\mathsf{S}_p\mathsf{P}^i + \frac{1}{\delta t}\mathsf{S}_z\mathsf{Z}$$

3. Solve an update-equation for the pressure

$$\mathsf{C}_1\delta\mathsf{P}_1^i = \mathsf{F}_P \tag{4.21}$$

$$\mathsf{C}_2\delta\mathsf{P}_2^i = \mathsf{F}_P \tag{4.22}$$

4. Update the new pressure

$$\mathsf{P}^{i+1} = \mathsf{P}^i + \delta\mathsf{P}_1^i + \delta\mathsf{P}_2^i$$

Since we are interested in the fractional step scheme that involves only one iteration the values at $i+1$ are associated with the values at $n+1$. For the prediction ($i=0$) we use values at n and we omit the indexes in $\delta \mathsf{P}_1$ and $\delta \mathsf{P}_2$. From the first step we can write

$$\frac{1}{\delta t}\mathsf{M}\left(\widetilde{\mathsf{U}}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\widetilde{\mathsf{U}}^{n+1}+\mathsf{G}\mathsf{P}^{n}=\mathsf{F}^{n+1}. \tag{4.23}$$

From (4.21) we have

$$\left(-\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{G}+\mathsf{S}_p\right)\left(\delta \mathsf{P}_1\right)+\mathsf{S}_p\mathsf{P}^{n}-\mathsf{S}_z\mathsf{Z}^{n+1}+\mathsf{D}\widetilde{\mathsf{U}}^{n+1,i+1}=\mathsf{0}. \tag{4.24}$$

Using the exact continuity equation

$$\mathsf{D}\mathsf{U}^{n+1}+\mathsf{S}_p\mathsf{P}^{n+1}-\mathsf{S}_z\mathsf{Z}^{n+1}=\mathsf{0} \tag{4.25}$$

and

$$\mathsf{P}^{n+1}=\mathsf{P}^{n}+\delta \mathsf{P}_1+\delta \mathsf{P}_2, \tag{4.26}$$

we can rewrite (4.24) as

$$\left(\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\right)\left(\delta \mathsf{P}_1\right)+\mathsf{S}_p\delta \mathsf{P}_2+\mathsf{D}\mathsf{U}^{n+1}-\mathsf{D}\widetilde{\mathsf{U}}^{n+1,i+1}=\mathsf{0},$$

that multiplied by $\frac{1}{\delta t}\mathsf{M}\mathsf{D}^{-1}$ gives

$$\mathsf{G}\left(\delta \mathsf{P}_1\right)+\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\widetilde{\mathsf{U}}^{n+1}\right)=-\frac{1}{\delta t}\mathsf{M}\mathsf{D}^{-1}\mathsf{S}_p\delta \mathsf{P}_2. \tag{4.27}$$

Equations (4.23,4.27 and 4.25) are the pressure stabilized rotational counterpart of (4.5,4.2 and 4.3).

Adding (4.23) and (4.27) and using (4.26) we obtain

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\widetilde{\mathsf{U}}^{n+1}+\mathsf{G}\mathsf{P}^{n+1}-\mathsf{G}\left(\delta \mathsf{P}_2\right)+\frac{1}{\delta t}\mathsf{M}\mathsf{D}^{-1}\mathsf{S}_p\delta \mathsf{P}_2=\mathsf{F}^{n+1}$$

which can be compared with the exact momentum equation,

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\mathsf{U}^{n+1}+\mathsf{G}\mathsf{P}^{n+1}=\mathsf{F}^{n+1},$$

to clarify the error we have introduced with the splitting. In the standard (non rotational, $\delta \mathsf{P}_2=0$) case we recover the original approximation (4.4). When the rotational form is

used, the approximation is $\mathsf{K}\mathsf{U}^{n+1} \approx \mathsf{K}\widetilde{\mathsf{U}}^{n+1} - \mathsf{G}\,(\delta\mathsf{P}_2) + \frac{1}{\delta t}\mathsf{M}\mathsf{D}^{-1}\mathsf{S}_p\delta\mathsf{P}_2$ where $\delta\mathsf{P}_2$ has been defined in (4.22).

Actually in our implementation we have omitted the term in the right hand side of (4.27). This implies perturbing the continuity equation (4.25) with $\mathsf{S}_p\delta\mathsf{P}_2$. Now the approximation used in the momentum equation is $\mathsf{K}\mathsf{U}^{n+1} \approx \mathsf{K}\widetilde{\mathsf{U}}^{n+1} - \mathsf{G}\,(\delta\mathsf{P}_2)$. Contrary to what happens in the usual pressure correction fractional step scheme our implementation introduces a perturbation in both the momentum and continuity equations and not only in the momentum equation.

Relation with other methods

Finally it is interesting to show, following [115], how some other well known schemes can also be described as pressure Schur complement techniques. Uzawa like iterations can be associated to the choice $\alpha_R = 0$. In fact for stationary calculations that is the choice Turek [115] recommends, with $\alpha_D \leqslant \mu$. SIMPLE like schemes can be associated to a Schur complement iteration when the preconditioner for the pressure Schur complement is taken as $\mathsf{C}^{-1} = -\mathsf{D}\widetilde{\mathsf{S}}^{-1}\mathsf{G}$, where $\widetilde{\mathsf{S}}$ is some approximation to S, for example its diagonal or the diagonal matrix obtained by summing its rows (if it does not lead to a zero diagonal).

Instead of using a Richardson preconditioned iteration to solve for (4.16) as in (4.17) a more elaborate preconditioned scheme, such as preconditioned GMRES, could be used. This leads to a method very similar to the one proposed in [42, 43, 68, 74]. The use of a GMRES iteration should make the method more robust and improve convergence compared to a Richardson iteration. We have implemented a preliminary version of the method proposed in [42,43,68,74] but very limited testing has been done and no conclusion can be drawn for the moment. Regarding the approximation of the inverse of the Schur complement for the pressure, C^{-1}, a more elaborate version was introduced in [74]. In the Stokes case it reduces to (4.18) but when the convective term is present it is supposed to improve the approximation. It reads

$$\mathsf{C}_*^{-1} = \mathsf{M}_P^{-1}\mathsf{A}_P\mathsf{T}^{-1} \tag{4.28}$$

where the matrices M_P^{-1} and T^{-1} are the ones introduced in (4.18). Actually in [42, 43, 68, 74] the cheaper approximation L^{-1} is used instead of T^{-1}. A_p is a discrete approximation to the convection-diffusion operator on the pressure finite element space that includes the terms used to stabilize the convective part,

$$\mathsf{A}_p = \frac{1}{\delta t}\mathsf{M}_p + \mathsf{K}_p(\mathsf{U}) + \mathsf{S}_c\left(\tau_1;\mathsf{U}\right).$$

K_p and S_c are given by

$$\mathsf{K}_p\left(\mathsf{U}\right)^{ab} = \left(N^a, \rho\ \boldsymbol{u}_h \cdot \nabla N^b\right) + \left(\nabla N^a, \mu \nabla N^b\right),$$

$$\mathsf{S}_c\left(\tau_1;\mathsf{U}\right)^{ab} = \left(\tau_1 \boldsymbol{u}_h \cdot \nabla N^a, \rho\ \boldsymbol{u}_h \cdot \nabla N^b\right),$$

where, as in Chapter 1, we denote the node indexes with superscripts a, b and the standard shape functions of node a by N^a. The approximation of the inverse of the Schur complement for the pressure, C_*^{-1} has been used in methods that converge to the monolithic solution at each time step. It could also be interesting to test it in fractional step like methods. Since C_*^{-1} is a better approximation than C^{-1} in the Navier Stokes case it could be used in (4.19) to obtain an enhanced fractional step scheme.

4.3 Velocity correction methods

In this Section the velocity correction pressure segregation methods based on a Discrete Pressure Poison Equation, as suggested in [6] will be introduced. They are called velocity corrector methods because it is the velocity, and not the pressure, that is extrapolated in the first step of the method. In [4] the appearance of velocity correction method is attributed to Guermond and Shen [52] but it can also be related to the scheme introduced by Karniadakis, Israeli and Orzag [67]. Moreover we would like to mention that an algorithm presented [44] and recommended in [48] has several similarities with the velocity correction method proposed in [6].

The main particularity of the method proposed in [4] is that it is obtained at the discrete level, as has been done for the pressure correction scheme. The continuity

equation is replaced by a discrete pressure Poisson equation obtained from the monolithic discretized problem. The predictor-corrector version is also presented.

4.3.1 The Discrete Pressure Poison Equation

The discrete pressure Poisson equation (DPPE) is obtained from the monolithic problem discretized both in space and time. The matrix form of the monolithic problem (1.10), supposing only homogeneous Dirichlet boundary conditions and neglecting the stabilization terms for a BDF1 time discretization reads:

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}+\mathsf{G}\mathsf{P}^{n+1}=\mathsf{F}^{n+1}, \tag{4.29}$$

$$\mathsf{D}\mathsf{U}^{n+1}=\mathsf{0}. \tag{4.30}$$

If the momentum equation (4.29) is multiplied by $\delta t\mathsf{D}\mathsf{M}^{-1}$ and the resulting equation is subtracted from the continuity equation (4.30), the DPPE is obtained

$$\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1}=\delta t\mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1}-\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}\right)+\mathsf{D}\mathsf{U}^{n}. \tag{4.31}$$

The system formed by (4.29, 4.31) is equivalent to the original monolithic discretized scheme (4.29, 4.30) and the boundary conditions arise naturally from the original scheme. There is no advantage in solving the coupled system that uses the DPPE equation directly. The advantage is that the segregation is now straight forward.

It is also interesting to point out that an alternative DPPE can be obtained if the momentum equation (4.29) is multiplied by $\delta t\mathsf{D}\mathsf{M}_L^{-1}$ instead of $\delta t\mathsf{D}\mathsf{M}^{-1}$ where M_L is the lumped mass matrix. It reads

$$\delta t\mathsf{D}\mathsf{M}_L^{-1}\mathsf{G}\mathsf{P}^{n+1}=\delta t\mathsf{D}\mathsf{M}_L^{-1}\left(\mathsf{F}^{n+1}-\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}-\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1}-\mathsf{U}^{n}\right)\right)+\mathsf{D}\mathsf{U}^{n+1}. \tag{4.32}$$

The system formed by (4.29, 4.32) is also equivalent to the original monolithic discretized scheme (4.29, 4.30). This does not happen if one naively approximates $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ by $\mathsf{D}\mathsf{M}_L^{-1}\mathsf{G}$ in (4.31). In that case, the system formed by (4.29) and the approximation to (4.31) is only an approximation to the original monolithic discretized scheme. We would like

to point this out because we had initially taken the naive approach. In that case, the predictor corrector scheme does not converge to the monolithic solution. Moreover, from the implementation point of view, we belive that (4.32) is also advantageous because it leads to the use of the same matrix, $\frac{1}{\delta t}\mathsf{M}+\mathsf{K}\left(\mathsf{U}^{n+1}\right)$, in both the DPPE and the momentum equation. Instead if (4.31) is used, in the DPPE only $\mathsf{K}\left(\mathsf{U}^{n+1}\right)$ appears.

As we discuss in Subsection 4.3.3, an extrapolation is needed for U^{n+1} to obtain the fractional step scheme. When $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ is used, the extrapolation is only needed for $\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}$. Instead if the lumped mass matrix is used, it is needed for both $\delta t\mathsf{D}\mathsf{M}_L^{-1}\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}$ and $\mathsf{D}\left(-\mathsf{M}_L^{-1}\mathsf{M}\mathsf{U}^{n+1}+\mathsf{U}^{n+1}\right)$. Our numerical experience indicates that the use of the extrapolation in the second term introduces no difficulties because M_L is a good approximation to M.

Before introducing the resulting fractional step and predictor corrector velocity correction schemes we will discuss the approximation of $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ (or $\mathsf{D}\mathsf{M}_L^{-1}\mathsf{G}$).

4.3.2 Approximation of $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$

When using continuous pressure interpolations, the construction of $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ is relatively expensive even if a diagonal mass matrix is used. Therefore, it is usually approximated as

$$\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\approx\mathsf{L},\qquad\text{with components}\quad \mathsf{L}^{ab}=-\left(\nabla N^a,\nabla N^b\right).$$

In [6] the following *enhanced* approximation is introduced

$$\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1}=\mathsf{L}\mathsf{P}^{n+1}+\left(\mathsf{D}\mathsf{M}^{-1}\mathsf{G}-\mathsf{L}\right)\mathsf{P}^{n+1}\approx\mathsf{L}\mathsf{P}^{n+1}+\left(\mathsf{D}\mathsf{M}^{-1}\mathsf{G}-\mathsf{L}\right)\widetilde{\mathsf{P}}_p^{n+1}\tag{4.33}$$

where $\widetilde{\mathsf{P}}_p^{n+1}$ is an extrapolation of order p of P^{n+1} obtained from previous known values in the case of the fractional step scheme. The DPPE then reads

$$\delta t\mathsf{L}\left(\mathsf{P}^{n+1}-\widetilde{\mathsf{P}}_p^{n+1}\right)=\delta t\mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1}-\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}-\mathsf{G}\widetilde{\mathsf{P}}_p^{n+1}\right)+\mathsf{D}\mathsf{U}^n.\tag{4.34}$$

In the predictor corrector case, in the first iteration $\widetilde{\mathsf{P}}_p^{n+1}$ will be used, but in the rest of the iterations the value from the previous iteration will be used. The DPPE then reads

$$\delta t\mathsf{L}\left(\mathsf{P}^{n+1,i+1}-\mathsf{P}^{n+1,i}\right)=\delta t\mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1}-\mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}-\mathsf{G}\mathsf{P}^{n+1,i}\right)+\mathsf{D}\mathsf{U}^n.\tag{4.35}$$

with $\mathsf{P}^{n+1,i} = \widetilde{\mathsf{P}}_p^{n+1}$ for $i = 0$. As we explain in the next Subsection, the velocity U^{n+1} is extrapolated from the values at previous time steps.

In the predictor corrector case, one could even decide to use a separate iterative loop for solving for the previous approximation, as we have already suggested for the convective term in the pressure correction scheme. In that case, the iteration index i would not correspond to the outer iterative loop for the uncoupling of the unknowns but to an internal iterative loop for solving iteratively the exact DPPE without needing to form $\mathsf{DM}^{-1}\mathsf{G}$. We could then rewrite (4.35) as using a Richardson iteration to solve for

$$\mathsf{DM}^{-1}\mathsf{G}\mathsf{P}^{n+1} = \mathsf{X}$$

with

$$\mathsf{X} = \mathsf{DM}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1}\right) + \frac{1}{\delta t}\mathsf{D}\mathsf{U}^{n}.$$

The iterative procedure would then read

$$\mathsf{L}\mathsf{P}^{n+1,i+1} = \mathsf{X} - \mathsf{DM}^{-1}\mathsf{G}\mathsf{P}^{n+1,i} + \mathsf{L}\mathsf{P}^{n+1,i}. \tag{4.36}$$

This iterative process allows us to solve for $\mathsf{DM}^{-1}\mathsf{GP}^{\mathsf{n+1}}$ without needing to assemble $\mathsf{DM}^{-1}\mathsf{G}$. Its efficiency, as compared to solving directly for $\mathsf{DM}^{-1}\mathsf{G}$ is something we will have to test numerically. Moreover, now one can use a non diagonal mass matrix, something that is not possible if one wants to solve directly for $\mathsf{DM}^{-1}\mathsf{G}$.

Seen as an iterative solver for $\mathsf{DM}^{-1}\mathsf{G}$, it is obvious that the previous approach can be applied not only to the predictor corrector scheme but also the fractional step scheme. If only one iteration is allowed, we recover the *enhanced* approximation suggested in [6] if $\mathsf{P}^{n+1,i=0} = \widetilde{\mathsf{P}}_p^{n+1}$ ($\mathsf{P}^{n+1,i=0} = \mathsf{P}^{n+1,j}$ in the predictor corrector case, where j is the outer loop for the uncoupling of the unknowns) and the usual one ($\mathsf{DM}^{-1}\mathsf{G} \approx \mathsf{L}$) if $\mathsf{P}^{n+1,0} = \mathsf{0}$.

We have discussed the approximation (or solution) of $\mathsf{DM}^{-1}\mathsf{G}$ when dealing with the velocity correction method but the same ideas can be applied to pressure correction methods. Obviously the same approximation (or iterative solution scheme) can be used for $\mathsf{DM}_L^{-1}\mathsf{G}$.

A more elaborate and robust option might be to solve for the discrete Laplacian with a conjugate gradient method using as preconditioner the continuous Laplacian.

4.3.3 Fractional step scheme

In order to obtain the fractional step scheme we start from the coupled system written with a DPPE (4.29, 4.31). The method is called a velocity correction method because it is the velocity that is extrapolated from values at previous time steps instead of the pressure. In the first step the pressure is obtained from the DPPE using an extrapolation of order q (denoted by $\widetilde{\mathsf{U}}_q^{n+1}$) of the velocity U^{n+1}. Then, U^{n+1} is obtained from the momentum equation (velocity correction step).

Using a BDF1 time discretization, the split scheme reads

$$\delta t \mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1} = \delta t \mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\widetilde{\mathsf{U}}_q^{n+1}\right)\widetilde{\mathsf{U}}_q^{n+1}\right) + \mathsf{D}\mathsf{U}^{n}, \tag{4.37}$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1} - \mathsf{U}^{n}\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} = \mathsf{F}^{n+1}. \tag{4.38}$$

Using approximation (4.33) for $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ we can obtain the following system:

$$\delta t \mathsf{L}\left(\mathsf{P}^{n+1} - \widetilde{\mathsf{P}}_p^{n+1}\right) = \delta t \mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\widetilde{\mathsf{U}}_q^{n+1}\right)\widetilde{\mathsf{U}}_q^{n+1} - \mathsf{G}\widetilde{\mathsf{P}}_p^{n+1}\right) + \mathsf{D}\mathsf{U}^{n},$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1} - \mathsf{U}^{n}\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} = \mathsf{F}^{n+1}.$$

A first order method in time can be obtained taking $q = p = 0$,

$$\delta t \mathsf{L}\mathsf{P}^{n+1} = \delta t \mathsf{D}\mathsf{M}^{-1}\mathsf{F}^{n+1} + \mathsf{D}\mathsf{U}^{n},$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1} - \mathsf{U}^{n}\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} = \mathsf{F}^{n+1}.$$

For a scheme with second order accuracy in time BDF2 time discretization and $q = p = 1$ must be used. The resulting system is then

$$\frac{2}{3}\delta t \mathsf{L}\left(\mathsf{P}^{n+1} - \mathsf{P}^{n}\right) = \frac{2}{3}\delta t \mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\mathsf{U}^{n}\right)\mathsf{U}^{n} - \mathsf{G}\mathsf{P}^{n}\right) + \mathsf{D}\left(\frac{4}{3}\mathsf{U}^{n} - \frac{1}{3}\mathsf{U}^{n-1}\right),$$

$$\frac{1}{2\delta t}\mathsf{M}\left(3\mathsf{U}^{n+1} - 4\mathsf{U}^{n} + \mathsf{U}^{n-1}\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} = \mathsf{F}^{n+1}.$$

The first remarkable fact about the previous equations is that despite we are using a velocity correction method which should be characterized by the fact that in going from one step to the next only an extrapolation for the velocity is used, actually extrapolations

for both the velocity and the pressure are used. The need for the pressure extrapolation comes from the use of an approximation of $\mathsf{DM}^{-1}\mathsf{G}$ (*enhanced* or not). The reason for needing the pressure extrapolation is therefore different to the reason for needing the velocity extrapolation (that is the true spirit of the velocity correction method) but, in any case, if the approximation of the discrete Laplacian is used, both are needed. Instead in the pressure correction method, only an extrapolation of the pressure is needed no matter whether the discrete Laplacian or an approximation to it is used.

Therefore, we can say that in order to obtain a 'pure' velocity correction scheme the discrete Laplacian must be used. As we have mentioned in the pressure correction case, extrapolations of order higher than one should help to reduce the splitting error but can cause instabilities. In [4] a third order method that used BDF3 time discretization and second order extrapolations for both the velocity and the pressure with the enhanced approximation for the discrete Laplacian was tested. Numerical experimentation showed that it was unstable as happens for third order pressure correction methods.

In this work we have solved the discrete Laplacian, directly or using a Richardson iteration, to obtain a 'pure' third order velocity correction method that only uses second order velocity extrapolation. This has allowed us to obtain a third order method that has shown to be stable in numerical examples we present at the end of this chapter. When the same cases are run with the third order method with an approximation of the discrete Laplacian used in [4] they are unstable (also if a pressure correction method is used). Moreover we have used the velocity correction BDF3 scheme with an approximation to the discrete Laplacian with a first order pressure extrapolation and second order velocity extrapolation. In that case, the third order accuracy is lost but stability is recovered. Therefore we can guess that the instability observed in [4] was caused by the use of second order pressure extrapolations and that different conclusions could have been drawn if a 'pure' velocity correction scheme had been used. Instabilities for third order velocity correction schemes are also observed in [50] where the fractional step scheme is obtained at the continuous level precluding the possibility of using a discrete Laplacian.

Second order velocity extrapolations may work better than second order pressure

extrapolations because the velocity satisfies an evolutionary equation; instead the pressure adapts itself instantaneously to satisfy the incompressibility constraint. From the convergence analysis of different pressure segregation methods it is known that the error estimates for the velocity are sharper than for the pressure [4]. Even in the monolithic case, where the order of the velocity error depends only on the time integration scheme used, pressure errors of order equal or higher than two cannot always be obtained for time integration schemes of order two or higher [56].

The third order accurate scheme used in the numerical examples is obtained by combining a BDF3 time discretization and a second order velocity extrapolation ($q = 2$). It reads

$$\frac{6}{11}\delta t \mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1} = \frac{6}{11}\delta t \mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\widetilde{\mathsf{U}}_q^{n+1}\right)\widetilde{\mathsf{U}}_q^{n+1}\right) + \mathsf{D}\left(\frac{18}{11}\mathsf{U}^n - \frac{9}{11}\mathsf{U}^{n-1} + \frac{2}{11}\mathsf{U}^{n-2}\right),$$

$$\frac{1}{6\delta t}\mathsf{M}\left(11\mathsf{U}^{n+1} - 18\mathsf{U}^n + 9\mathsf{U}^{n-1} - 2\mathsf{U}^{n-2}\right) + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} = \mathsf{F}^{n+1}.$$

Although we do not have an analytical proof but only numerical evidence, this is one of the few [67, 84] third order schemes in which velocity and pressure are segregated of which we are aware.

4.3.4 Predictor corrector scheme

Starting from the coupled system where mass conservation is enforced by the DPPE (4.29, 4.31) the obtention of the predictor corrector scheme arises naturally. No additional terms have to be introduced as happens in the pressure correction case. Denoting by a superscript i the ith iteration of the scheme, the resulting predictor corrector method for a BDF1 time discretization and Picard linearization of the convective term is:

$$\delta t \mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1,i+1} = \delta t \mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i}\right) + \mathsf{D}\mathsf{U}^n,$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1,i+1} - \mathsf{U}^n\right) + \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1} + \mathsf{G}\mathsf{P}^{n+1,i+1} = \mathsf{F}^{n+1}.$$

$\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$ can be approximated or solved as we have already discussed. If approximation (4.35) is used we obtain

$$\delta t \mathsf{L}\left(\mathsf{P}^{n+1,i+1} - \mathsf{P}^{n+1,i}\right) = \delta t \mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1} - \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i} - \mathsf{G}\mathsf{P}^{n+1,i}\right) + \mathsf{D}\mathsf{U}^n,$$

$$\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1,i+1}-\mathsf{U}^{n}\right)+\mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1}+\mathsf{G}\mathsf{P}^{n+1,i+1}=\mathsf{F}^{n+1}.$$

The method has to be properly initialized, preferably starting the process with a splitting error at least of the same order as the time discretization. For the previous equations $\mathsf{U}^{n+1,0}=\widetilde{\mathsf{U}}_q^{n+1}$ and $\mathsf{P}^{n+1,0}=\widetilde{\mathsf{P}}_p^{n+1}$, with $p=q=0$ can be used but a better convergence is obtained if a second order splitting ($p=q=1$) is used [4].

The second order method, using BDF2 time discretization, reads

$$\frac{2}{3}\delta t\mathsf{L}\left(\mathsf{P}^{n+1,i+1}-\mathsf{P}^{n+1,i}\right)=\frac{2}{3}\delta t\mathsf{D}\mathsf{M}^{-1}\left(\mathsf{F}^{n+1}-\mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i}-\mathsf{G}\mathsf{P}^{n+1,i}\right)+\mathsf{D}\left(\frac{4}{3}\mathsf{U}^{n}-\frac{1}{3}\mathsf{U}^{n-1}\right),$$

$$\frac{1}{2\delta t}\mathsf{M}\left(3\mathsf{U}^{n+1,i+1}-4\mathsf{U}^{n}+\mathsf{U}^{n-1}\right)+\mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1}+\mathsf{G}\mathsf{P}^{n+1,i+1}=\mathsf{F}^{n+1},$$

with appropriate initializations $\mathsf{U}^{n+1,0}=\widetilde{\mathsf{U}}_q^{n+1}$ and $\mathsf{P}^{n+1,0}=\widetilde{\mathsf{P}}_p^{n+1}$. As in the pressure correction case for the predictor corrector scheme high order extrapolation can be used because the stability is not compromised. When the iterative procedure converges the solution of the monolithic system is recovered.

If the discrete Laplacian with lumped mass matrix is used, as in (4.32), the BDF2 velocity correction scheme reads

$$\begin{aligned}\frac{2}{3}\delta t\mathsf{D}\mathsf{M}_L^{-1}\mathsf{G}\mathsf{P}^{n+1,i+1} &= \frac{2}{3}\delta t\mathsf{D}\mathsf{M}_L^{-1}\left(\mathsf{F}^{n+1}-\mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i}\right)\\ &\quad+\mathsf{D}\mathsf{M}_L^{-1}\mathsf{M}\left(-\mathsf{U}^{n+1,i}+\frac{4}{3}\mathsf{U}^{n}-\frac{1}{3}\mathsf{U}^{n-1}\right)+\mathsf{D}\mathsf{U}^{n+1,i},\end{aligned}$$

$$\frac{1}{2\delta t}\mathsf{M}\left(3\mathsf{U}^{n+1,i+1}-4\mathsf{U}^{n}+\mathsf{U}^{n-1}\right)+\mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1}+\mathsf{G}\mathsf{P}^{n+1,i+1}=\mathsf{F}^{n+1}.$$

Now only $\mathsf{U}^{n+1,0}$ needs to be initialized.

Instead of treating the nonlinearity with the same loop as the coupling of the variables an internal loop could be used for the convective nonlinearity, as has been mentioned in the pressure correction scheme. In that case the term corresponding to matrix K in the momentum equation would be replaced by $\mathsf{K}\left(\mathsf{U}^{n+1,i+1}\right)\mathsf{U}^{n+1,i+1}$ and an internal loop would be used to solve it. As in the pressure correction case, we have implemented nested loops with the linearization of the convective term treated in the inner loop in order to have a method that allows us to recover the usual predictor corrector version without

nested loops and the fractional step scheme. For the velocity correction case we have also implemented the alternative that deals with the convective nonlinearity in the outer loop. This version resembles the monolithic version more closely and it is cheaper because the matrix needs to be assembled only once per outer iteration.

4.3.5 Stabilized Scheme

In this section we will present the stabilized version of the problem, using split OSS stabilization, a BDF1 time discretization and the matrices introduced in Chapter 1. The matrix version of the problem reads as follows:

$$
\begin{aligned}
&\left(\delta t \mathsf{D}\mathsf{M}^{-1}\mathsf{G} - \mathsf{S}_{\mathsf{p}}\left(\tau_1^{n+1,i}\right)\right)\mathsf{P}^{n+1,i+1} \\
&\quad = \delta t \mathsf{D}\mathsf{M}^{-1}\Big[\mathsf{F}^{n+1} - \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i} \\
&\quad -\mathsf{S}_u\left(\tau_1^{n+1,i};\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i} + \mathsf{S}_y\left(\tau_1^{n+1,i};\mathsf{U}^{n+1,i}\right)\mathsf{Y}^{n+1,i} \\
&\quad -\mathsf{S}_d\left(\tau_2^{n+1,i}\right)\mathsf{U}^{n+1,i} + \mathsf{S}_w\left(\tau_2^{n+1,i}\right)\mathsf{w}^{n+1,i}\Big] \\
&\quad +\mathsf{D}\mathsf{U}^n - \mathsf{S}_z\left(\tau_1^{n+1,i}\right)\mathsf{Z}^{n+1,i},
\end{aligned}
$$

$$
\begin{aligned}
&\frac{1}{\delta t}\mathsf{M}\left(\mathsf{U}^{n+1,i+1} - \mathsf{U}^n\right) + \mathsf{K}\left(\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1} + \mathsf{G}\mathsf{P}^{n+1,i+1} \\
&\quad +\mathsf{S}_u\left(\tau_1^{n+1,i};\mathsf{U}^{n+1,i}\right)\mathsf{U}^{n+1,i+1} - \mathsf{S}_y\left(\tau_1^{n+1,i};\mathsf{U}^{n+1,i}\right)\mathsf{Y}^{n+1,i} \\
&\quad +\mathsf{S}_d\left(\tau_2^{n+1,i}\right)\mathsf{U}^{n+1,i+1} - \mathsf{S}_w\left(\tau_2^{n+1,i}\right)\mathsf{w}^{n+1,i} = \mathsf{F}^{n+1},
\end{aligned}
$$

$$
\mathsf{M}_\pi \mathsf{Y}^{n+1,i+1} - \mathsf{C}\left(\mathsf{U}^{n+1,i+1}\right)\mathsf{U}^{n+1,i+1} = 0,
$$

$$
\mathsf{M}_\pi \mathsf{Z}^{n+1,i+1} - \mathsf{G}_\pi \mathsf{P}^{n+1,i+1} = 0,
$$

$$
\mathsf{M}_\pi \mathsf{W}^{n+1,i+1} - \mathsf{D}\mathsf{U}^{n+1,i+1} = 0.
$$

As in the pressure correction case, we have included the terms corresponding to $\tau_2 \neq 0$ that have been neglected in [4] because they help to enforce incompressibility, something that is particularly important for two fluid flows. Approximation (4.35) can be introduced to avoid dealing with $\mathsf{D}\mathsf{M}^{-1}\mathsf{G}$.

Since the fractional step scheme can be seen as the first iteration of the predictor corrector scheme we only present the stabilized version for the P-C case. The stabilized fractional step schemes can be found in [4] using both split and non-split OSS. In the first iteration the velocity is extrapolated from values at previous time steps. This extrapolation is used not only for the viscous and convective term, but also for the stabilization terms associated to the momentum and continuity equations. Further the projection array Y is also extrapolated for $i = 0$. When the approximation (4.35) for $\mathsf{DM}^{-1}\mathsf{G}$ is used we also set $\mathsf{P}^{n+1,0} = \widetilde{\mathsf{P}}_p^{n+1}$.

From the theoretical point of view, to obtain a scheme of order r a time discretization a scheme of order r must be used together with an extrapolation of the velocity (and of the pressure when a continuous Laplacian is used) of order $r-1$. From the practical point of view one can use extrapolations of order higher than $r-1$ to reduce the splitting error. In the fractional step version the extrapolations must be chosen so that the scheme is stable. In the predictor corrector case no restriction exists and they should be chosen so as to minimize the number of iterations of the method. In our simulations we have typically used $p = q = 1$ for the BDF1 time discretization and $p = 1$, $q = 2$ for the BDF2 case both for the fractional step and predictor corrector schemes.

4.3.6 Remarks on the ASGS and non split OSS stabilized cases

The use of the velocity correction scheme with ASGS or non split OSS stabilization shows some particularities, specially when combined with the enriched pressure two phase model presented in Chapter 2. The ASGS stabilized case is presented but the same observations apply to the non split OSS stabilized case. In order to simplify the presentation the stabilization terms related to τ_2 are neglected because they show no special behavior when ASGS or non split OSS stabilization are used . Moreover the presentation is restricted to linear elements to avoid the inclusion of viscous terms in the stabilization. In Chapter 1 only the matrix version of the problem for the split OSS stabilized case had been presented so the first step is to introduce the matrix version of the monolithic problem using ASGS

stabilization that reads

$$\frac{1}{\delta t}\mathsf{M}\mathsf{U}^{n+1} + \mathsf{K}\left(\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{G}\mathsf{P}^{n+1} + \mathsf{S}_u\left(\tau_1;\mathsf{U}^{n+1}\right)\mathsf{U}^{n+1} + \mathsf{S}_{up}\left(\tau_1;\mathsf{U}^{n+1}\right)\mathsf{P}^{n+1}$$
$$= \mathsf{F}^{n+1} + \mathsf{S}_{uf}\left(\tau_1;\mathsf{U}^{n+1}\right) + \frac{1}{\delta t}\mathsf{M}\mathsf{U}^{n},$$

$$\mathsf{D}\mathsf{U}^{n+1} + \mathsf{S}_{pu}\left(\tau_1\right)\mathsf{U}^{n+1} + \mathsf{S}_p\left(\tau_1\right)\mathsf{P}^{n+1} = \mathsf{S}_{pf}\left(\tau_1\right).$$

The use of ASGS stabilization introduces some new matrices and vectors that are defined as follows

$$\mathsf{S}_{up}\left(\tau_1;\mathsf{U}^{n+1}\right)_i^{ab} = \left(\tau_1 \boldsymbol{u}_h^{n+1}\cdot\nabla N^a, \partial_i N^b\right),$$

$$\mathsf{S}_{uf}\left(\tau_1;\mathsf{U}^{n+1}\right)_i^{a} = \left(\tau_1 \boldsymbol{u}_h^{n+1}\cdot\nabla N^a, f_i + (\rho/\delta t)\ u_i^n\right),$$

$$\mathsf{S}_{pu}\left(\tau_1\right)_j^{ab} = \left(\tau_1\partial_j N^a, \rho\ \boldsymbol{u}_h^{n+1}\cdot\nabla N^b + (\rho/\delta t)\ N^b\right),$$

$$\mathsf{S}_{pf}\left(\tau_1;\mathsf{U}^{n+1}\right)^{a} = \left(\tau_1\partial_i N^a, f_i + (\rho/\delta t)\ u_i^n\right).$$

Moreover S_u is redefined to include the transient terms

$$\mathsf{S}_u\left(\tau_1;\mathsf{U}^{n+1}\right)_{ij}^{ab} = \left(\tau_1 \boldsymbol{u}_h^{n+1}\cdot\nabla N^a, \rho\ \boldsymbol{u}_h^{n+1}\cdot\nabla N^b + (\rho/\delta t)\ N^b\right)\delta_{ij}.$$

The DPPE for the can now be obtained by multiplying the momentum equation by $\delta t\mathsf{D}\mathsf{M}^{-1}$ and substracting the continuity equation. It reads

$$\left[\delta t\mathsf{D}\mathsf{M}^{-1}\left(\mathsf{G}+\mathsf{S}_{up}\right) - \mathsf{S}_p\right]\mathsf{P}$$
$$= -\mathsf{S}_{pf} + \mathsf{S}_{pu}\mathsf{U} + \delta t\mathsf{D}\mathsf{M}^{-1}\left[\mathsf{F}^{n+1} + \mathsf{S}_{uf} + \frac{1}{\delta t}\mathsf{M}\mathsf{U}^n - \left(\mathsf{K}+\mathsf{S}_u\right)\mathsf{U}\right], \tag{4.39}$$

where the dependencies of stabilization matrices and vectors on τ and U^{n+1} have been eliminated to simplify the notation. In [4] the term $\mathsf{S}_{up}\mathsf{P}$ is sent to the right hand side to end up with

$$\left[\delta t\mathsf{D}\mathsf{M}^{-1}\mathsf{G} - \mathsf{S}_p\right]\mathsf{P}$$
$$= -\mathsf{S}_{pf} + \mathsf{S}_{pu}\mathsf{U} + \delta t\mathsf{D}\mathsf{M}^{-1}\left[\mathsf{F}^{n+1} + \mathsf{S}_{uf} + \frac{1}{\delta t}\mathsf{M}\mathsf{U}^n - \left(\mathsf{K}+\mathsf{S}_u\right)\mathsf{U} - \mathsf{S}_{up}\widetilde{\mathsf{P}}^{n+1}\right].$$

One advantage of doing this is that a symmetric matrix is obtained. The disadvantage is that despite we are using a velocity correction method we need an extrapolation for

pressure ($\widetilde{\mathsf{P}}^{n+1}$) even if a discrete Laplacian is used. We have not found any problems in using the approach proposed in [4] in the examples shown in this Chapter.

In order to combine the velocity correction scheme with the enriched pressure two phase model presented in Chapter 2, equation (4.39) has been preferred. The reason for doing so is that for the pressure extrapolation, $\widetilde{\mathsf{P}}^{n+1}$, the pressure from the previous step is used. When an enriched pressure is used this would introduce important complications. The pressure enrichment at step n would be needed when solving for step $n+1$. A correct integration would then imply combining the enhanced integration from step n with the enhanced integration from step $n+1$. In order to avoid these complications in the examples presented in Chapter 5 equation (4.39) has been used.

4.4 Open boundary conditions

In the previous sections we have supposed mainly Dirichlet velocity boundary conditions as is usually done in the literature. In this section we will try to clarify what to do with the pressure on Neumann velocity boundaries, Γ_{nu}. In [48] such boundaries are classified into open boundaries and traction boundaries. The term open boundaries is reserved exclusively for Neumann velocity boundaries that arise from the fact of having to cut the domain in order to be able to perform a simulation, for example, the outflow of the domain when simulating the flow behind a cylinder. Those Neumann velocity boundaries that are present in the real problem, such as a free surface, are called traction boundaries. On both of them $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{t}$. The difference is that on traction boundaries the value of $\boldsymbol{t}$ is known from the physical problem but on open boundaries some approximation is needed. Several options are discussed in [48] for such an approximation. The fact of not knowing the value for $\boldsymbol{t}$ gives some freedom on the choices to use in the open boundaries case. In any case, it will always be an approximation. In the monolithic case written in divergence form the open boundary condition usually used is $\boldsymbol{n} \cdot \boldsymbol{\sigma} = \boldsymbol{0}$. Since we want our predictor corrector versions to converge to the monolithic solution we will prefer our segregated versions also to satisfy that condition. Therefore the same treatment will be favored in

both cases (open or traction boundaries).

In fractional step schemes obtained at the continuous level the pressure is prescribed to zero on $\Gamma_{\rm nu}$. Actually in [50] it is pointed out that in the incremental version it is the pressure increment that is prescribed to zero on the Neumann velocity boundary. That is, the pressure is prescribed to the value from the previous time step on $\Gamma_{\rm nu}$. Then the pressures on $\Gamma_{\rm nu}$ for all time steps coincide with the initial one that is usually zero. When the approximation $\mathsf{DM}^{-1}\mathsf{G} \approx \mathsf{L}$ is used, the scheme obtained at the discrete is also forced to satisfy $p = 0$ on $\Gamma_{\rm nu}$. Then it is evident that even in a predictor corrector case (for example 4.9, 4.10) the solution will not be able to coincide with that of the monolithic system if there are open boundaries. If at the open boundary we have $\boldsymbol{n}\cdot\boldsymbol{\sigma} = \mathbf{0}$, projecting on the normal direction we get $\boldsymbol{n}\cdot\boldsymbol{\sigma}\cdot\boldsymbol{n} = 0$ and therefore $p = 2\ \mu\ n_i\ \partial_j u_i\ n_j$, that in the monolithic case can be different from zero. One possible solution is to use $p = 2\ \mu\ n_i\ \partial_j u_i\ n_j$ on $\Gamma_{\rm nu}$ instead of $p = 0$ when solving the equation for the pressure in the segregated case [101].

When the only approximation introduced in $\mathsf{DM}^{-1}\mathsf{G}$ is to use a diagonal mass matrix the equation for the pressure can be solved without any boundary condition as is done in [115]. Then when the iterative procedure converges one recovers exactly the solution of the monolithic system.

As we have already mentioned, in [4] the approximation $\mathsf{DM}^{-1}\mathsf{G} \approx \mathsf{L}$ has been enhanced by using (4.34) in the fractional step case and (4.35) in the predictor corrector case. Even in that case, when solving the system, the pressure has been imposed to zero on $\Gamma_{\rm nu}$. As in the case when $\mathsf{DM}^{-1}\mathsf{G} \approx \mathsf{L}$ is used, such prescription precludes the possibility of reaching the monolithic solution when there are open boundaries. The first solution that comes to the mind is not to fix any prescription on the pressure, but if that is done, the pressure would be undefined up to an additive constant, something that does not happen in the monolithic case with open boundaries. The solution we propose will be presented for the preconditioned Richardson iteration to solve $\mathsf{DM}^{-1}\mathsf{G}$ we have suggested, taking as a starting point the enhanced approximation proposed by [4], but is also applicable when only one iteration is done and the approximation is recovered.

The preconditioned Richardson iteration used to solve

$$\mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1} = \mathsf{X}$$

in the case with no Neumann velocity boundary conditions (4.36) is modified only in the definition of the preconditioning matrix L (that is replaced by L^*). In the case with open boundaries it reads

$$\mathsf{L}^*\mathsf{P}^{n+1,i+1} = \mathsf{X} - \mathsf{D}\mathsf{M}^{-1}\mathsf{G}\mathsf{P}^{n+1,i} + \mathsf{L}^*\mathsf{P}^{n+1,i}.$$

The matrix L^* is obtained by modifying matrix L in the degrees of freedom corresponding to open boundaries as is usually done when the pressure is prescribed. The difference is the right hand side is not altered. In this way the pressure is not prescribed to any value. Only the preconditioner for the Richardson iteration is altered. We have used this strategy even when only one Richardson iteration is done (the discrete Laplacian is approximated by the continuous one) and have found satisfactory results for both the pressure correction and velocity correction schemes. We believe that prescribing the pressure or its increment to zero when the discrete approach is used to obtain the segregation follows what is done when the splitting is obtained at the continuous level. The small modification we propose seems more natural when the scheme is obtained at the discrete level.

4.5 Numerical examples

In this section we present three numerical examples to compare the velocity correction and pressure correction schemes introduced in this Chapter. The results are also compared against the solution obtained with a monolithic solver. BDF1, BDF2 and BDF3 time discretizations are used. For BDF2 and BDF3 discretizations second order velocity extrapolation is used. In the velocity correction fractional step schemes, when used with the discrete Laplacian, this lead to a third order splitting error. For the pressure, first order extrapolation is used for all time discretizations. It will be shown that second order pressure extrapolations not only makes the pressure correction scheme unstable as is well known but also the velocity correction scheme with continuous Laplacian approximation.

In the first example the evolution of the flow to the steady state in a driven cavity will be used to compare the two schemes. In the second example the flow behind a cylinder, the most classical example for transient flow, is analyzed. It is shown that the third order VC scheme with discrete Laplacian remains stable when the other third order schemes diverge. Finally a convergence test is performed to show that the velocity correction scheme allows to obtain third order accuracy in the L2 norm of the velocity. For the first and third examples Split OSS stabilization is used and for the second one ASGS is used.

In order to simplify the presentation of the examples the following nomenclature is used. The pressure correction and velocity correction schemes are denoted 'PC' and 'VC' respectively. Fractional step versions are denoted by 'FS'. To identify the predictor corrector scheme we shall use 'Imon' because 'PC' has already been used for the pressure correction scheme. In our examples we use both the discrete Laplacian (with Lumped mass matrix) denoted by 'LD' and its approximation by the continuous Laplacian denoted by 'LC'.

4.5.1 Driven Cavity

The first example is a cavity flow problem at Reynold number $Re = 100$. The domain is a unit square discretized with a 21×21 triangular mesh. The velocity is prescribed to zero at the bottom and lateral boundaries and to (1,0) at the top boundary and upper corners (leaky lid).

Despite we are dealing with a stationary problem we have chosen to run a transient calculation from an initial zero velocity on all of the nodes except those on the upper boundary. The time step used is $\delta t = 1.0$. The goal is to test the evolution of the proposed methods towards the stationary solution and their numerical dissipation.

VC - PC comparison using a fractional step scheme with continuous Laplacian

Figures 4.1 and 4.2 show the transient residual evolution using VC and PC fractional step schemes with continuous Laplacian and both BDF1 and BDF2 time discretizations.

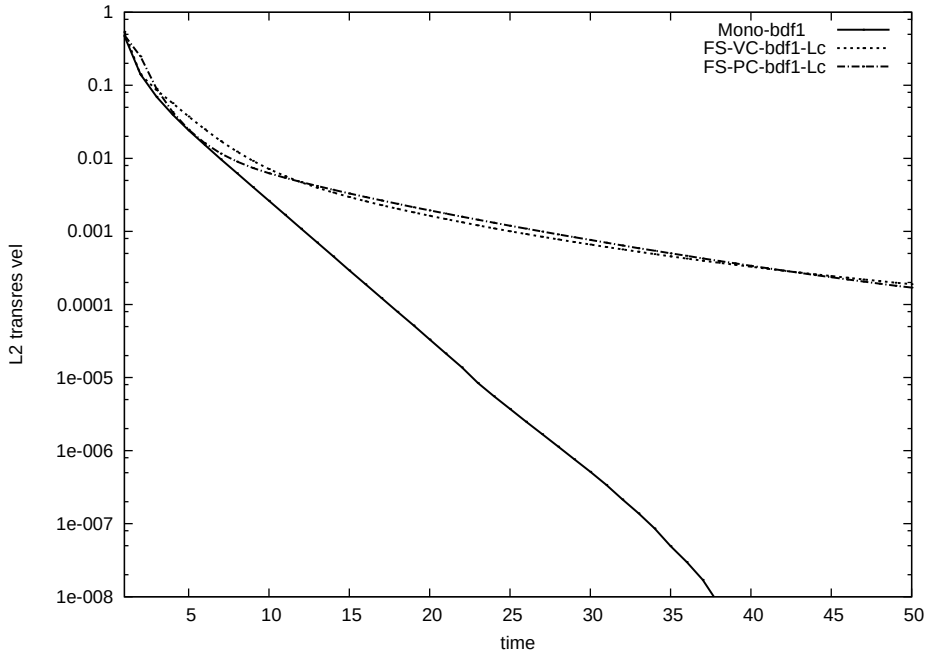


Figure 4.1: Transient residual BDF1 - PC vs. VC using a continuous Laplacian

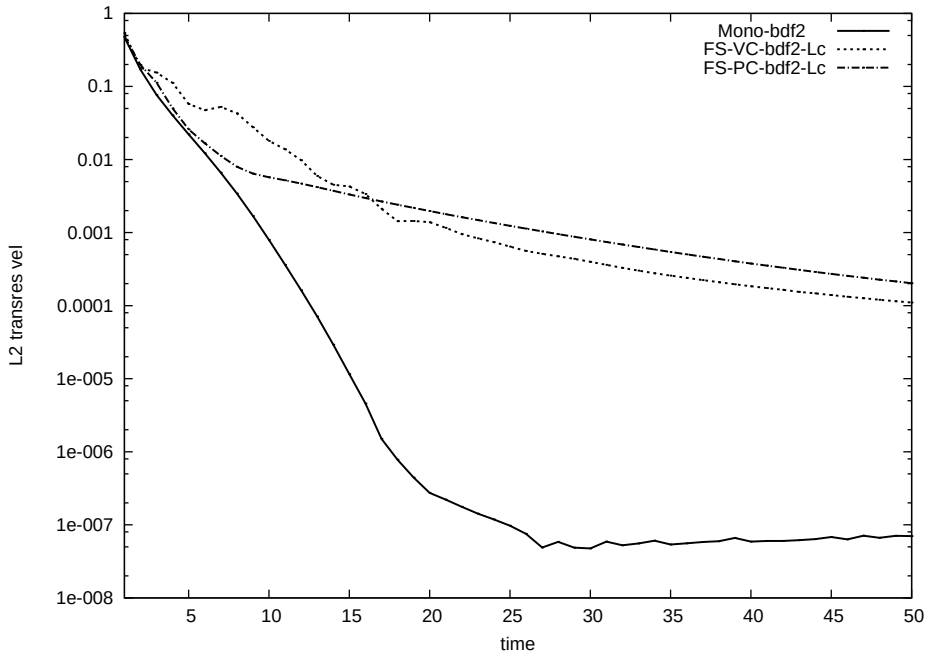


Figure 4.2: Transient residual BDF2 - PC vs. VC using a continuous Laplacian

The objective is to compare VC and PC schemes using a dissipation argument based on the evolution of the transient residual. With the first order time discretization no significant difference is observed between the VC and PC schemes. For BDF2 during the first 20 seconds the PC method follows the monolithic solver closer than the VC method but after that the opposite is observed.

In the previous examples it can also be interesting to analyze the L2 error for both the velocity and the pressure obtained by comparing with the monolithic solution (Figures 4.3 to 4.6). The L2 pressure error is much smaller when a VC scheme is used with both time integration schemes. For the velocity the PC scheme shows some slight advantage when the first order method is used. In the BDF2 case, after some few steps the VC scheme shows smaller velocity errors. Therefore we can conclude that (at least for this case) the VC scheme provides better results.

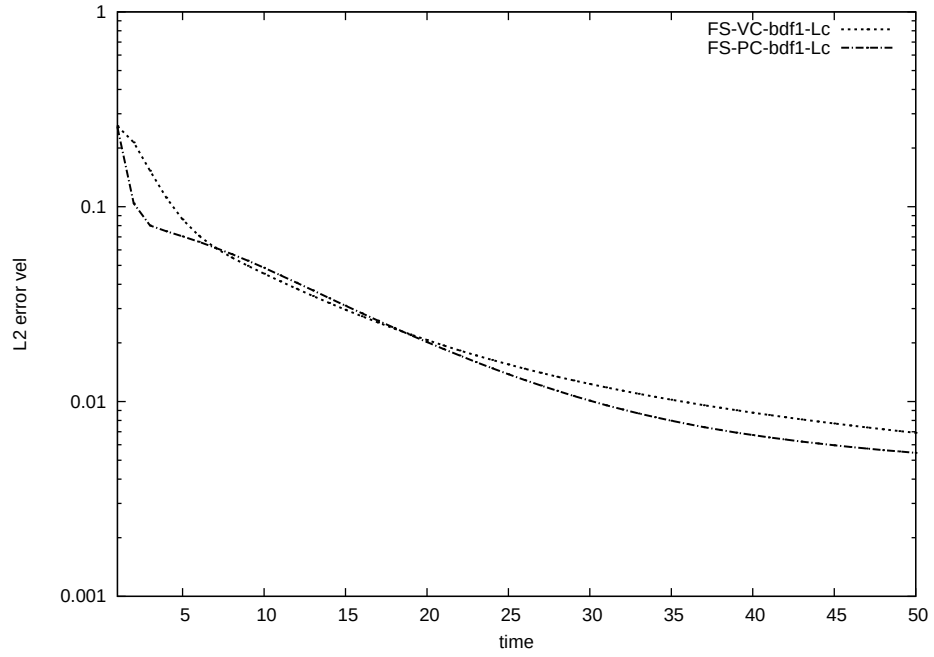


Figure 4.3: Velocity error BDF1 - PC vs. VC using a continuous Laplacian

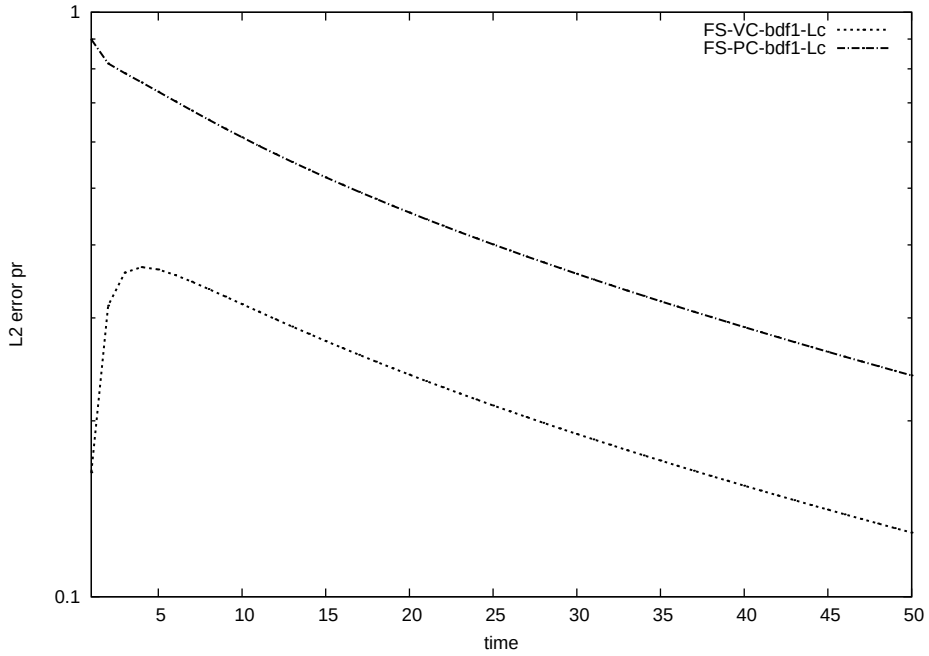


Figure 4.4: Pressure error BDF1 - PC vs. VC using a continuous Laplacian

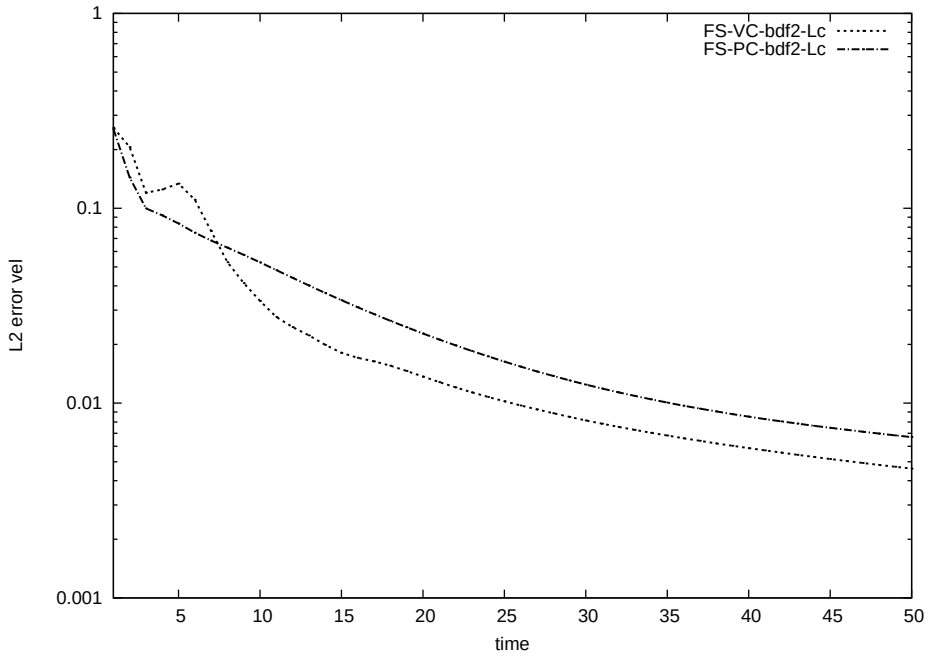


Figure 4.5: Velocity error BDF2 - PC vs. VC using a continuous Laplacian

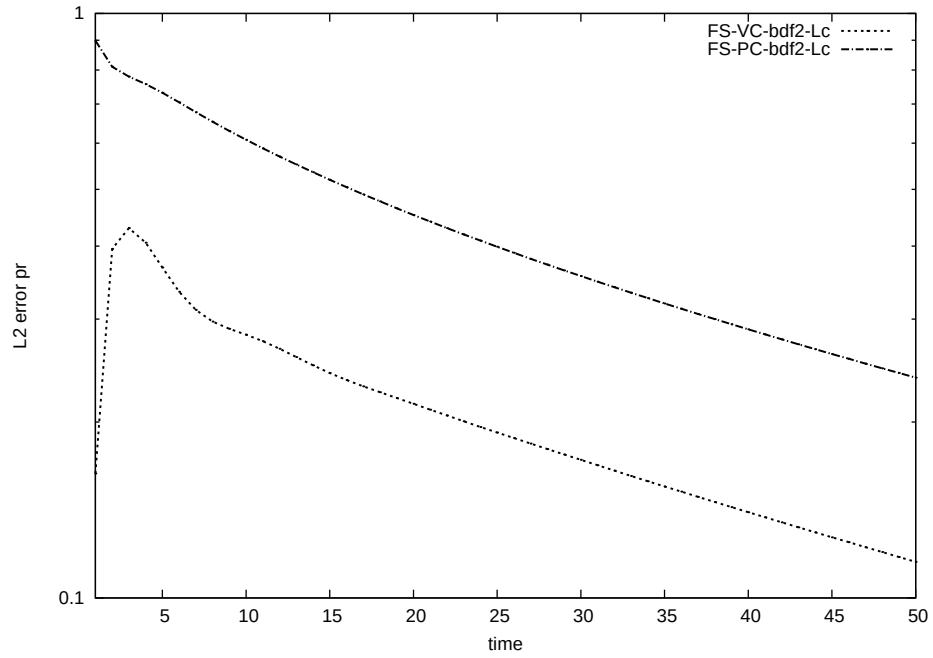


Figure 4.6: Pressure error BDF2 - PC vs. VC using a continuous Laplacian

LC - LD comparison using PC and VC fractional step schemes

In this subsection the main objective is to analyze the influence of using a continuous or discrete Laplacian. A fractional step scheme with both pressure and velocity correction versions will be used. As in the previous case, our first indicator is the evolution of the transient residual.

We first analyze the results obtained with the pressure correction scheme. When the first order time integrator is used, very little difference can be observed between using a continuous or discrete Laplacian (Figure 4.7). With the second order time integrator the discrete Laplacian provides better results (Figure 4.8).

Our second indicator is the L2 error for both the velocity and the pressure. Using the BDF1 scheme, the discrete Laplacian provides better results for the velocity error (Figure 4.9). The advantage is much more notorious for the pressure error (Figure 4.10). For both the velocity and the pressure error the advantage of using the discrete Laplacian shows up more clearly for the second order scheme as shown in Figures 4.11 to 4.12 .

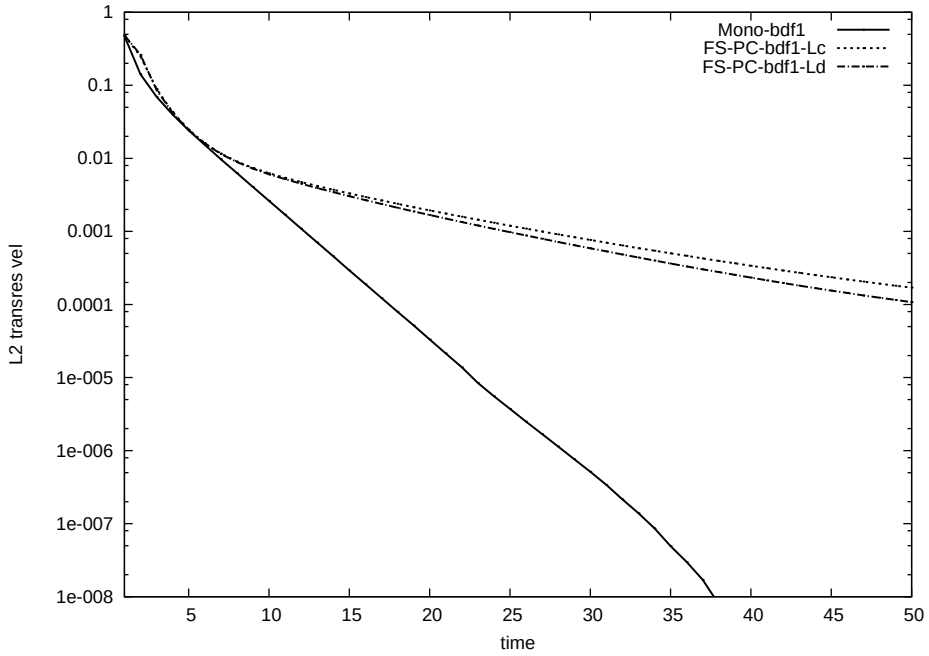


Figure 4.7: Transient residual BDF1 - Lc vs. Ld using the PC scheme

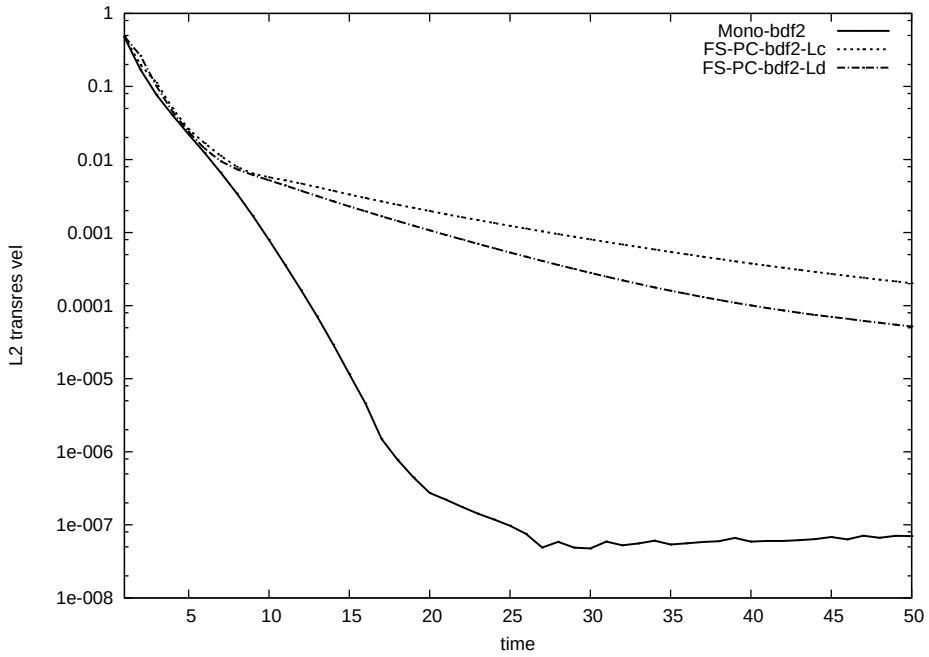


Figure 4.8: Transient residual BDF2 - Lc vs. Ld using the PC scheme

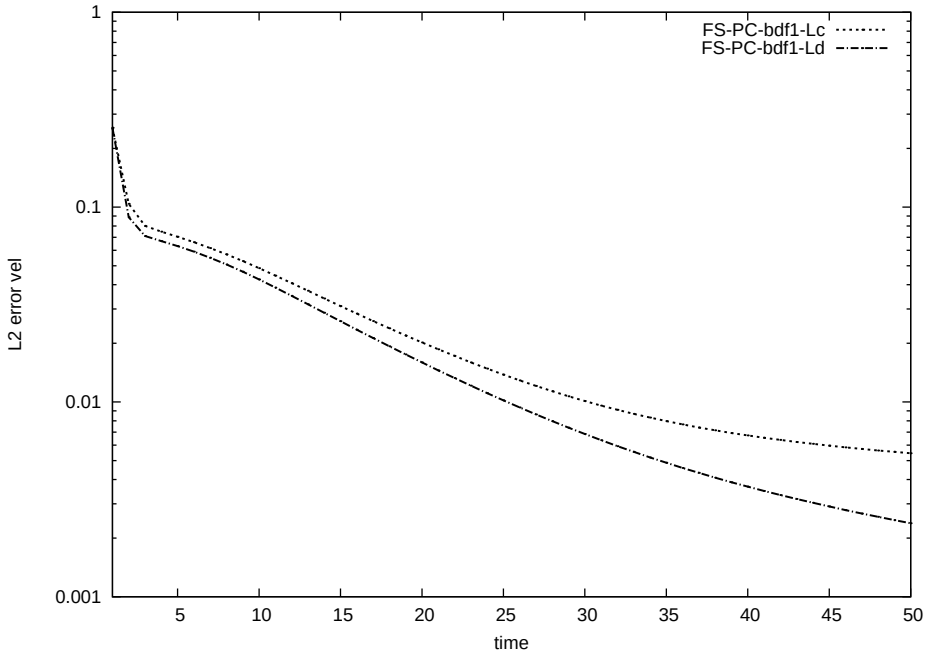


Figure 4.9: Velocity error BDF1 - Lc vs. Ld using the PC scheme

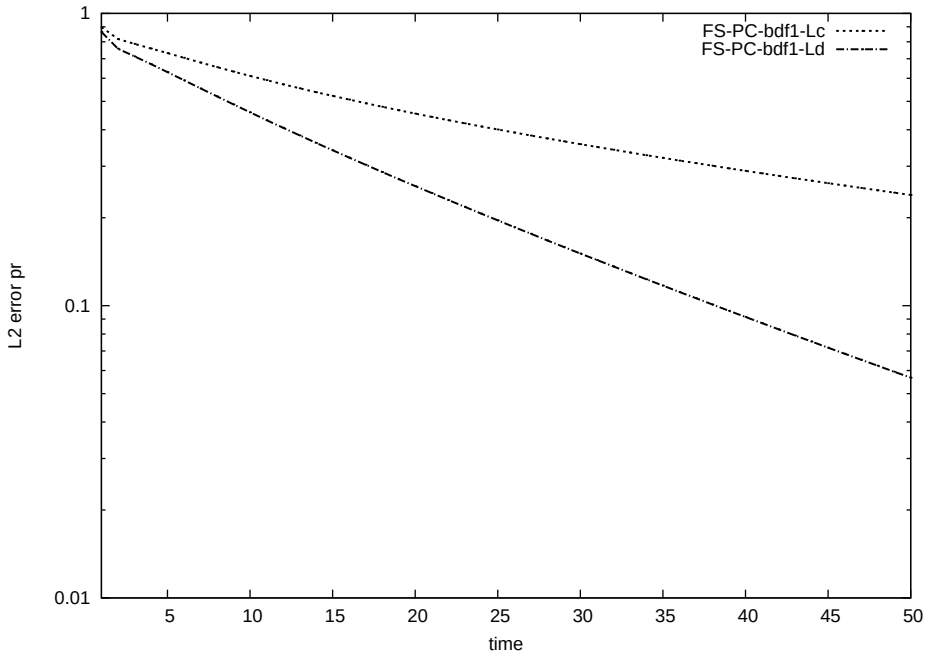


Figure 4.10: Pressure error BDF1 - Lc vs. Ld using the PC scheme

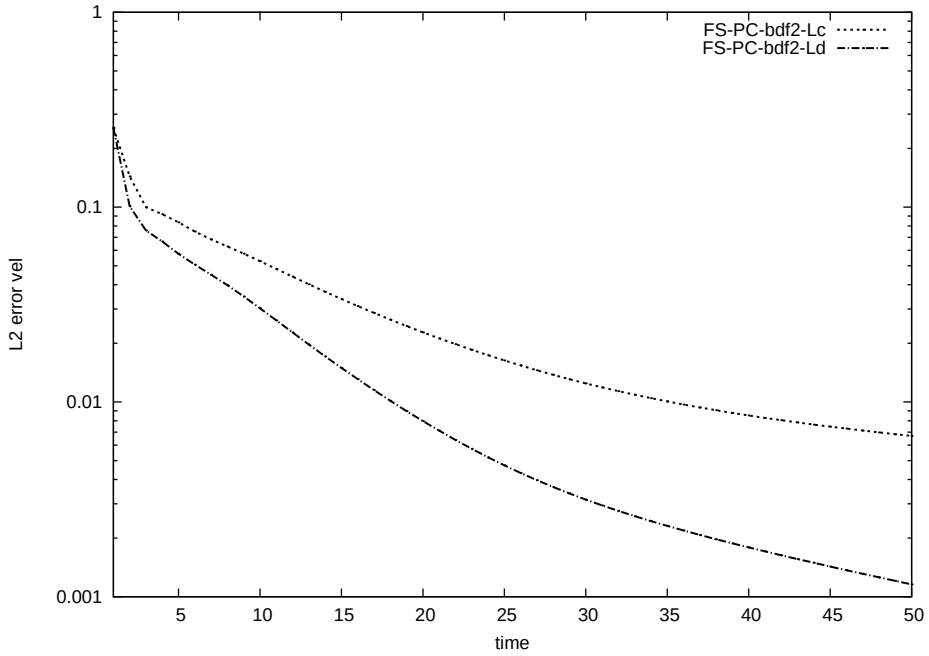


Figure 4.11: Velocity error BDF2 - Lc vs. Ld using the PC scheme

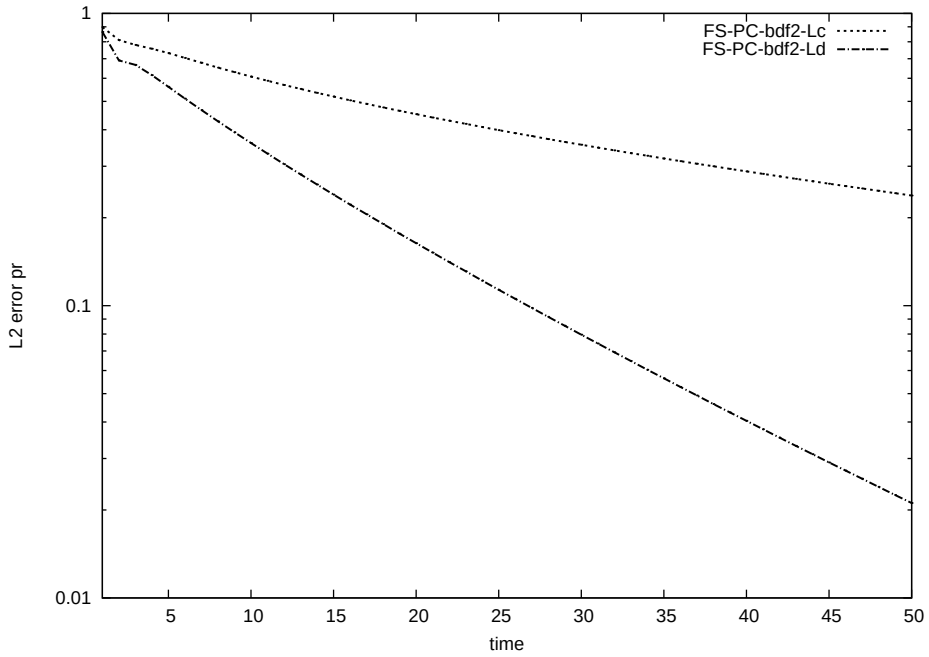


Figure 4.12: Pressure error BDF2 - Lc vs. Ld using the PC scheme

In the velocity correction case, using BDF1, again very little difference is observed in the evolution of the transient residual between the continuous and discrete Laplacians (Figure 4.13). Using BDF2, contrary to what we would expect, the transient residual is reduced more rapidly when the continuous Laplacian is used (Figure 4.14).

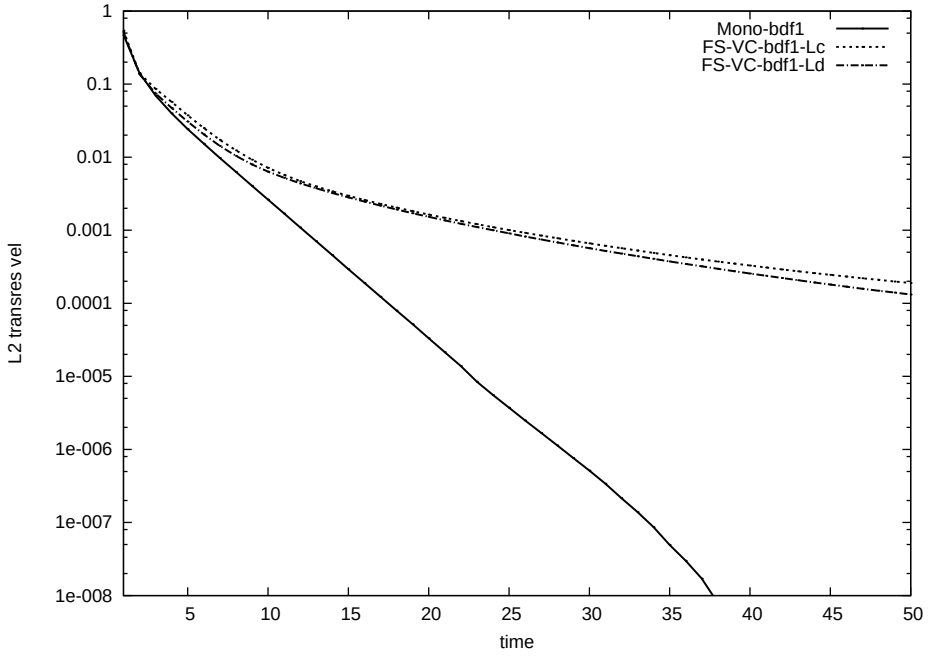


Figure 4.13: Transient residual BDF1 - Lc vs. Ld using the VC scheme

As in the pressure correction case, for the first order scheme the discrete Laplacian provides better results for the velocity error and specially for the pressure error (Figures 4.15 and 4.16).

With the second order scheme, up to $t = 35s$ the continuous Laplacian provides smaller velocity errors but after that the discrete Laplacian works better (Figure 4.17). For the pressure error, the results are significantly improved when the discrete Laplacian is used (Figure 4.18). This somehow contradicts what we had observed for the evolution of the transient residual in the BDF2 VC case. Which should be a better indicator of the goodness of a method, the evolution of the transient residual or the pressure error, is not so clear to us.

Despite this is a very small example and the computational times are not very

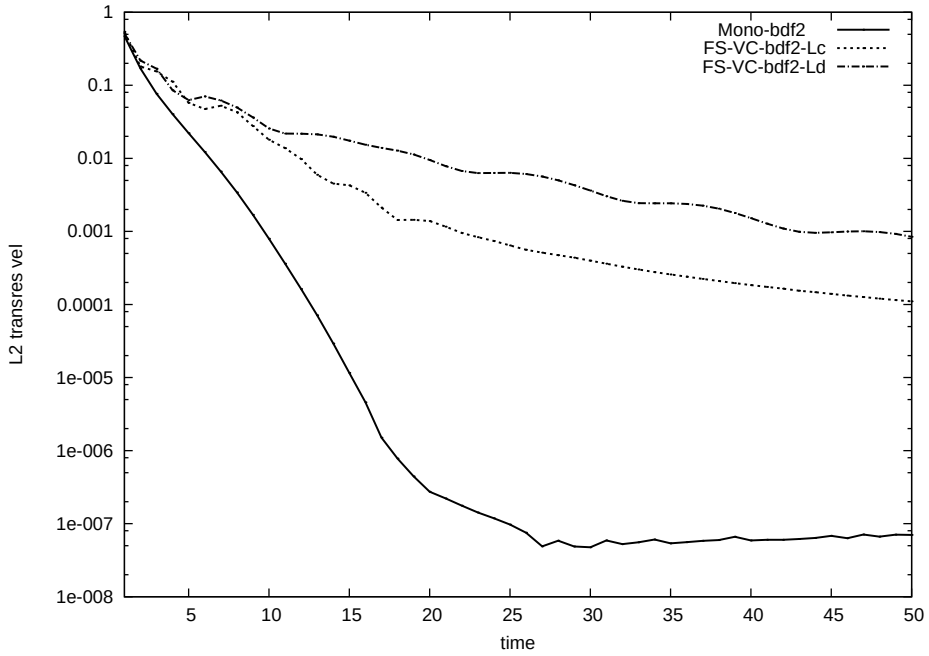


Figure 4.14: Transient residual BDF2 - Lc vs. Ld using the VC scheme

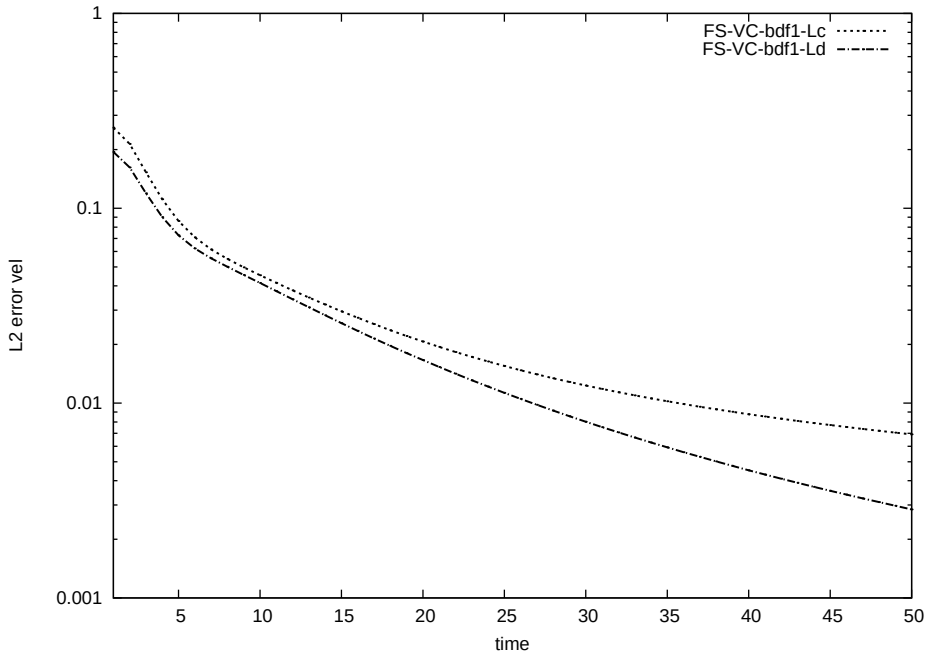


Figure 4.15: Velocity error BDF1 - Lc vs. Ld using the VC scheme

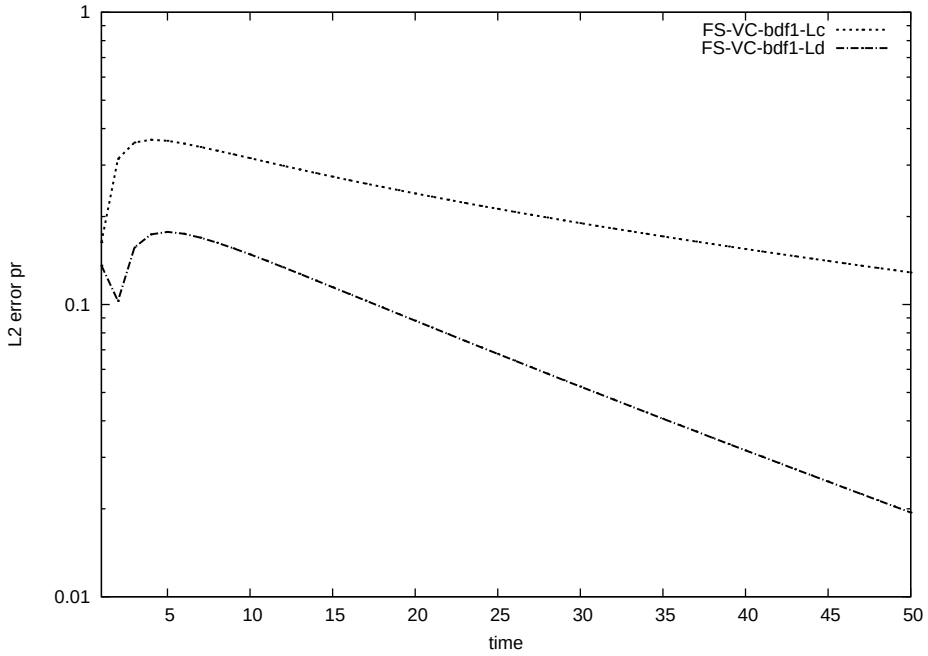


Figure 4.16: Pressure error BDF1 - Lc vs. Ld using the VC scheme

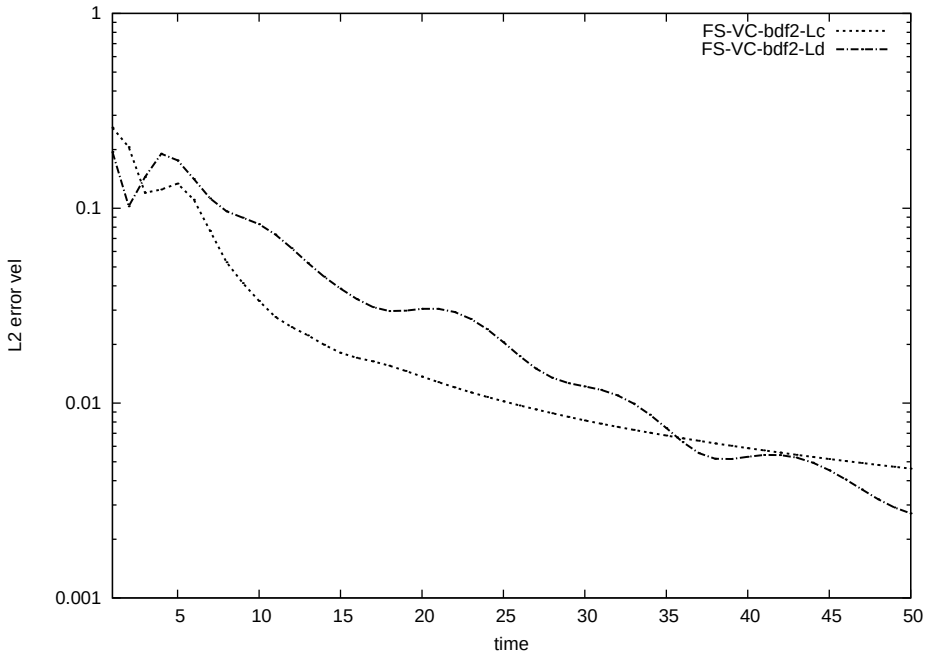


Figure 4.17: Velocity error BDF2 - Lc vs. Ld using the VC scheme

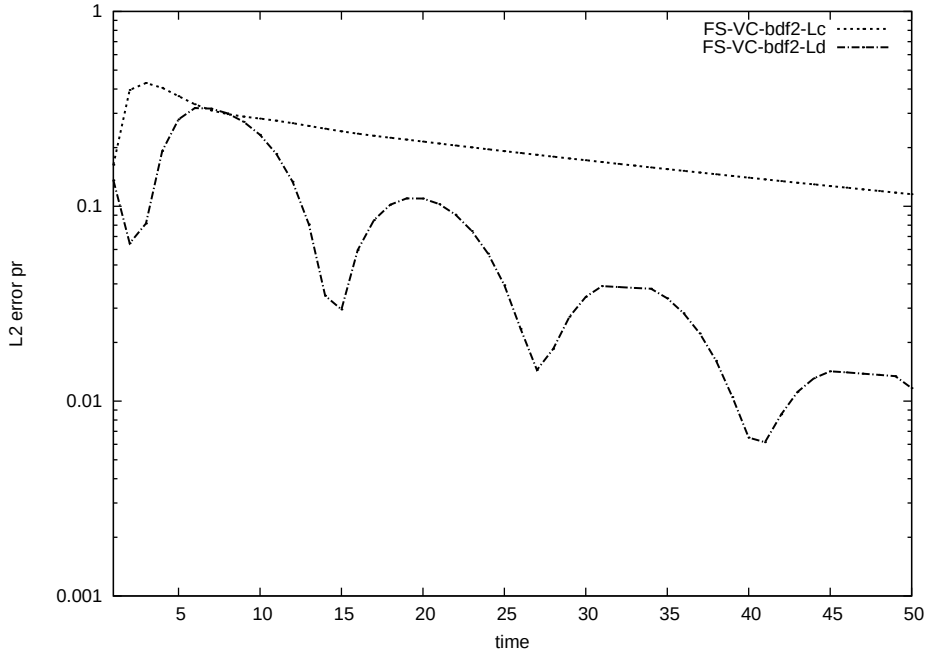


Figure 4.18: Pressure error BDF2 - Lc vs. Ld using the VC scheme

significative in Table 4.1 we present the total CPU times for VC and PC schemes using both Laplacians. More significative results are presented in the cylinder example. The use of the discrete or continuous Laplacian has very little influence for this example. The PC scheme takes more time than the VC scheme. This is related to the fact that using the VC scheme the convective non linearity takes less iterations to converge.

	PC	VC
L cont	$9.58s$	$7.19s$
Ldisc	$9.83s$	$6.98s$

Table 4.1: Total cpu time for the fractional step schemes

VC - PC comparison using the predictor corrector scheme with LC

Since predictor corrector methods converge to the monolithic solution the most important result for comparison purposes is the number of iterations per time step. The velocity correction scheme works significantly better that the pressure correction version specially in the BDF2 case, as can be seen in Figures 4.19 and 4.20. We have used the usual predictor corrector scheme where no nested loops are used. A maximum of 20 predictor corrector iterations per time step with a tolerance of 10^{-5} based on the variation of the L2 norm of the velocity divided by the norm of the velocity has been used. Between 5 to 8 time steps have been need so that the predictor corrector iterations fall bellow the maximum permitted limit of 20 iterations.

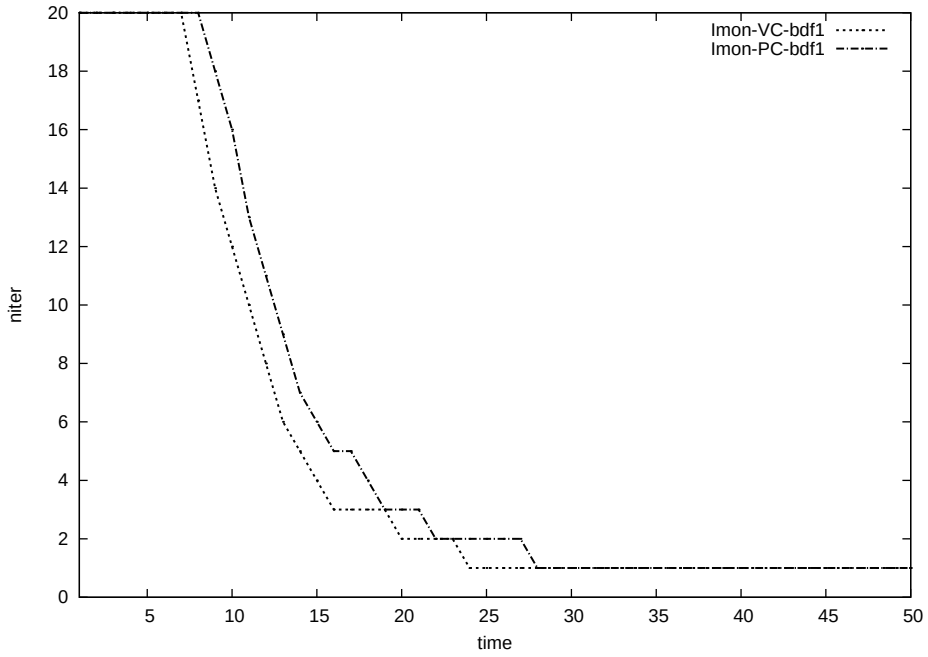


Figure 4.19: Number of iterations BDF1

Regarding the transient residual and the velocity error VC and PC work similarly. The behavior of the pressure velocity errors does not provide much information. Except for the pressure error with the BDF1 time discretization where the VC scheme shows lower errors, in the rest of the cases the PC scheme shows smaller errors specially in the second half of the run as can be seen in Figures 4.21 to 4.24 . This should not be

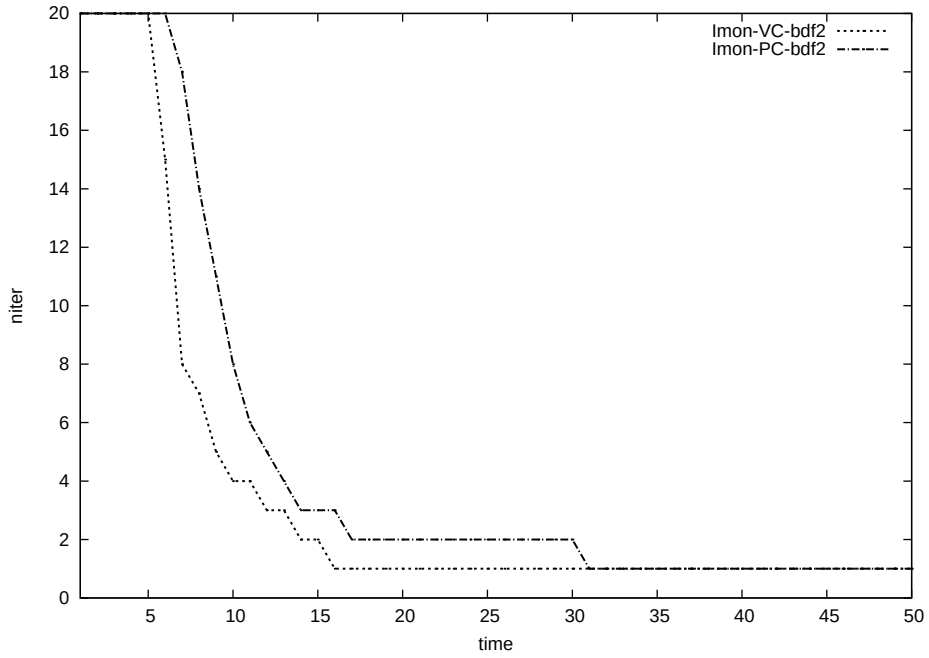


Figure 4.20: Number of iterations BDF2

interpreted as a real advantage of the PC scheme because it is probably related to the fact that it is doing more iterations per time step. Therefore the comparison of the velocity and pressure errors does not seem to be an interesting indicator in the predictor corrector case.

Richardson iteration for the discrete Laplacian

Two options have been proposed to solve the discrete Laplacian in this Chapter. The first one is the straightforward use of the conjugate gradient algorithm. The second one is to use a Richardson iteration preconditioned with the continuous Laplacian. The choice between the two options is based on a cost argument represented by the CPU time per time step. The results for both the VC and PC fractional step schemes are shown in Figures 4.27 and 4.28. The straightforward use of the CG algorithm proves more efficient with both schemes.

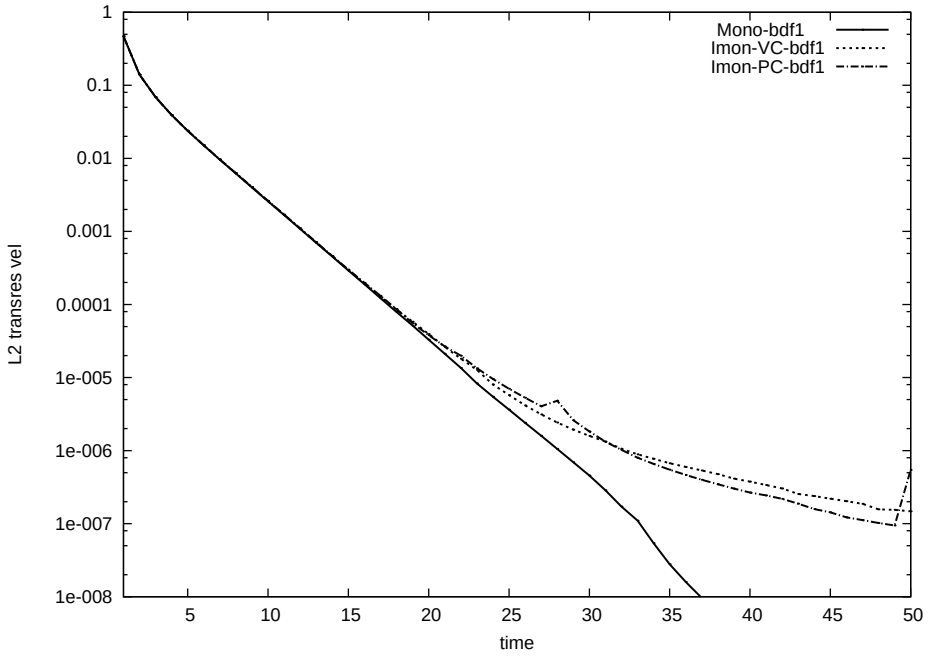


Figure 4.21: Transient residual BDF1

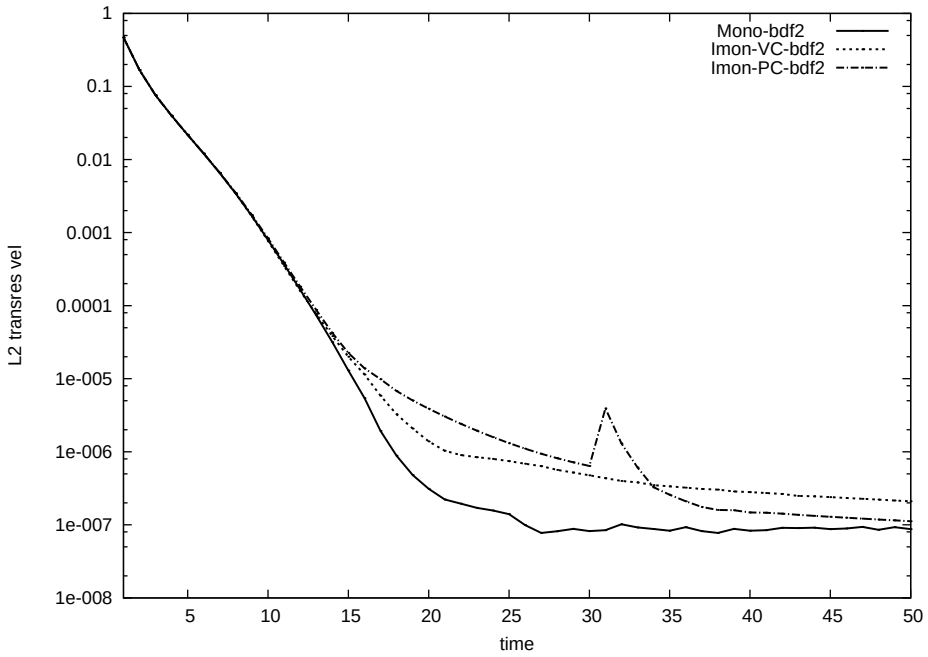


Figure 4.22: Transient residual BDF2

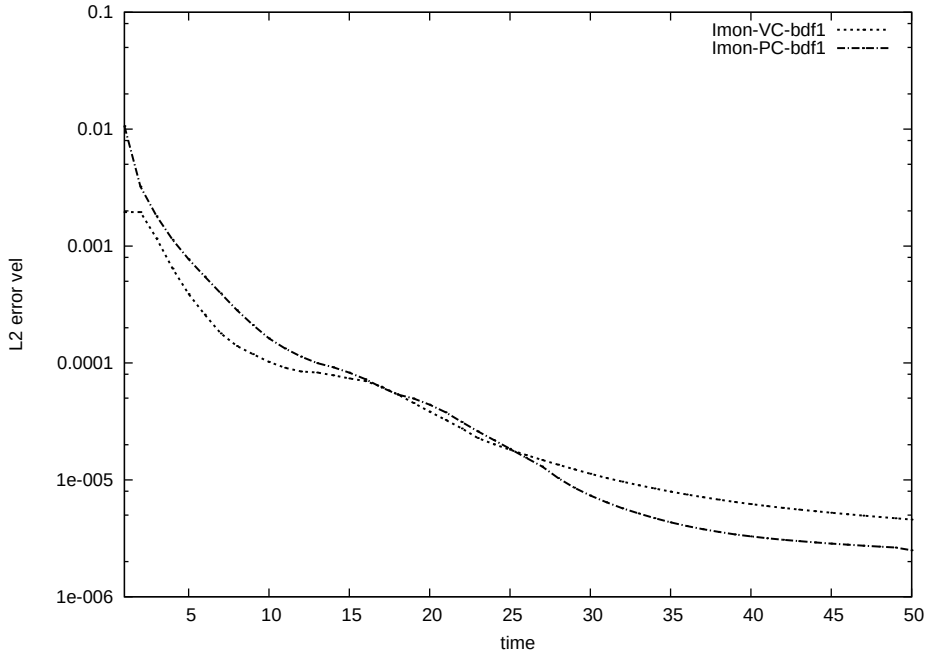


Figure 4.23: Velocity error BDF1

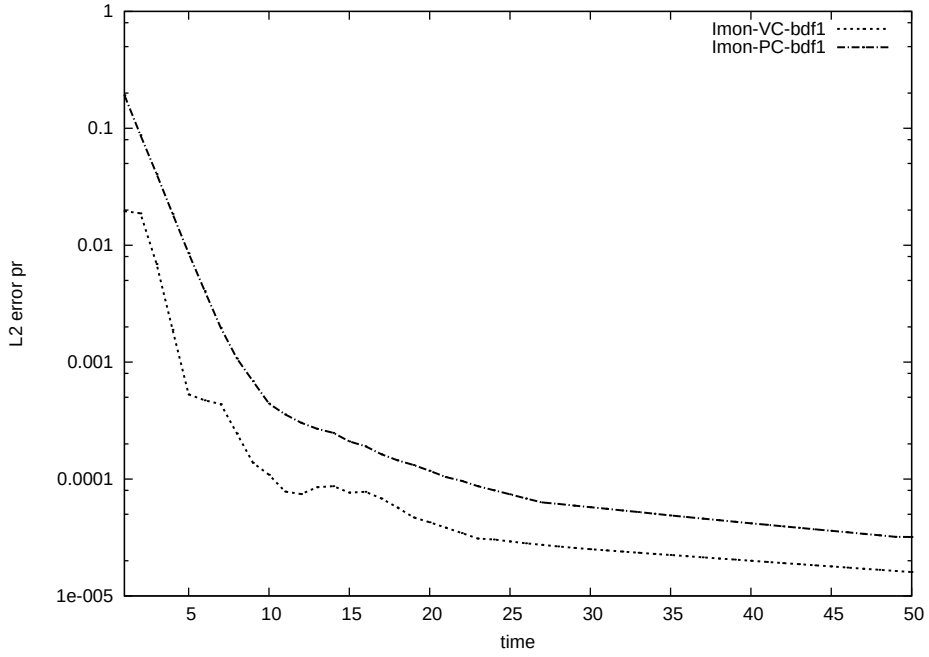


Figure 4.24: Pressure error BDF1

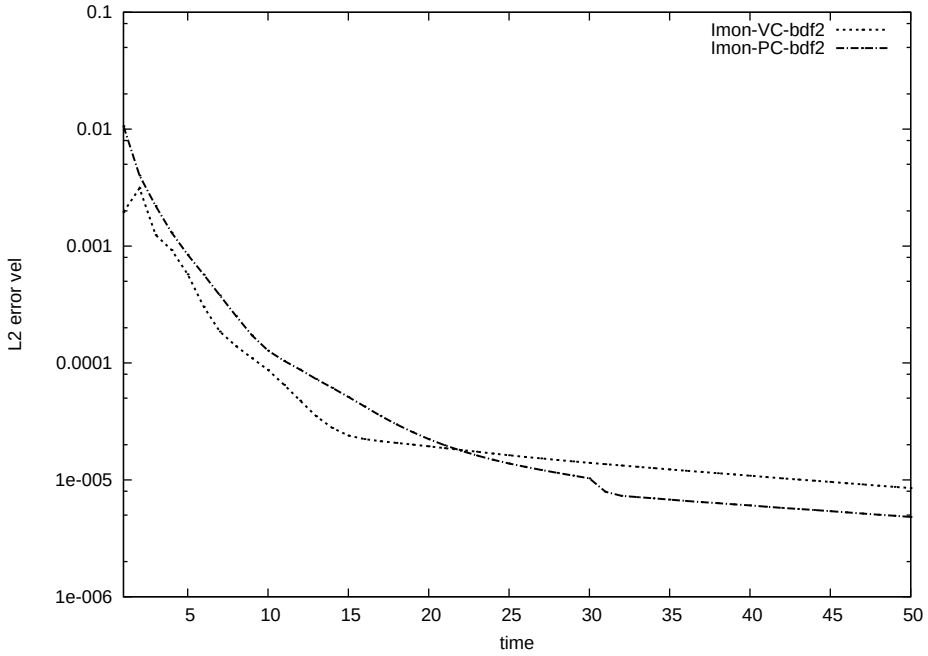


Figure 4.25: Velocity error BDF2

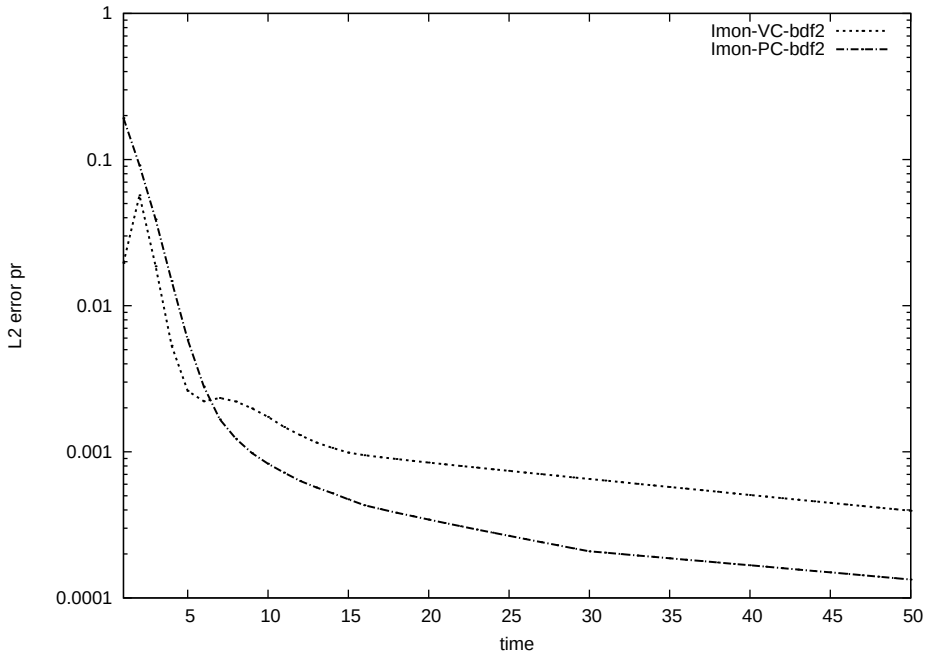


Figure 4.26: Pressure error BDF2

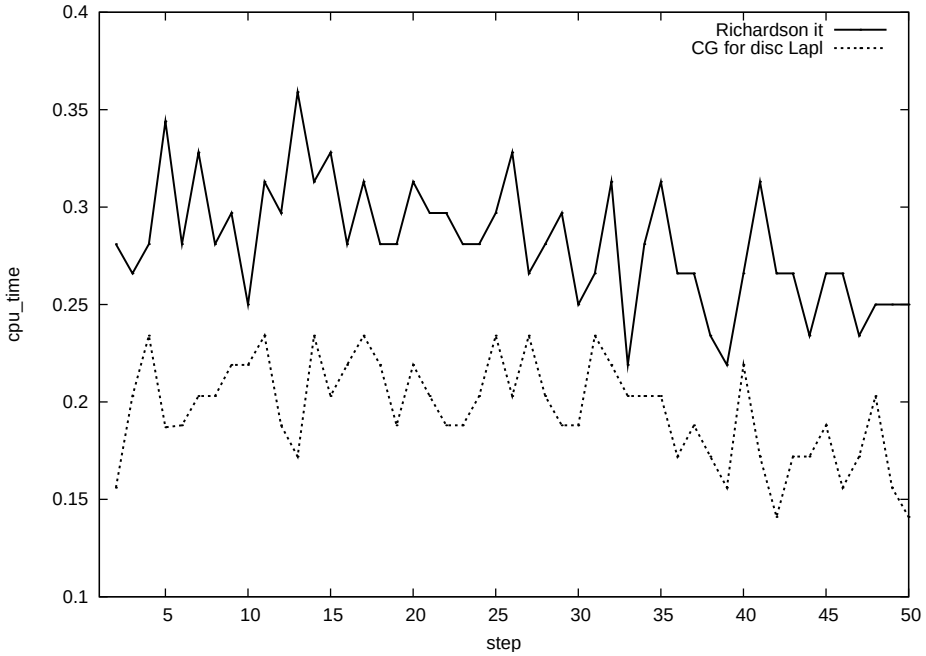


Figure 4.27: CPU time per time step - PC case

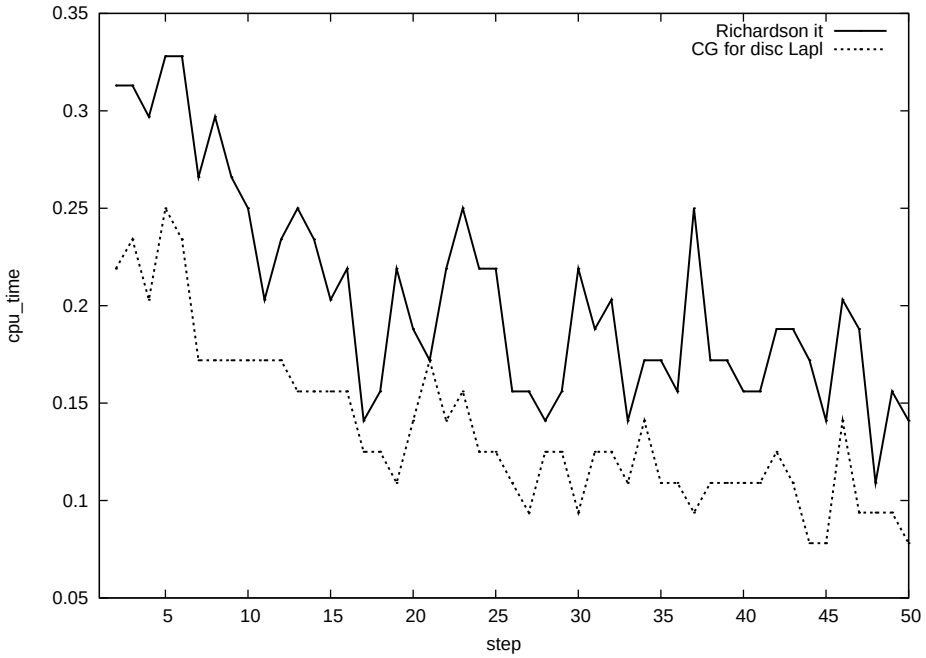


Figure 4.28: CPU time per time step - VC case

4.5.2 Flow behind a cylinder

The second example is the flow behind a cylinder at Reynold number $Re = 190$. This example is essentially 2-D but it has been run with a 3-D mesh. The mesh, provided by Professor Rainald Lohner, has a special placement of points in the vicinity of the cylinder [75]. It is formed by 108147 tetrahedral linear elements and 30000 nodes. In Figure 4.29 the surface mesh is shown. The computational domain is $\Omega = [0, 19] \times [0, 8] \times [0, 0.2] \setminus D$, with the cylinder D of diameter 1 centered at $(4, 4)$. The velocity at $x = 0$ is prescribed to $(1, 0, 0)$. At $y = 0$, $y = 8$, $z = 0$ and $z = 0.2$ the normal component of the velocity is set to zero and the tangential components are left free. At the outflow ($x = 19$) zero traction is prescribed. The time step is $\delta t = 0.05$ and the total time is $t = 100.0$.

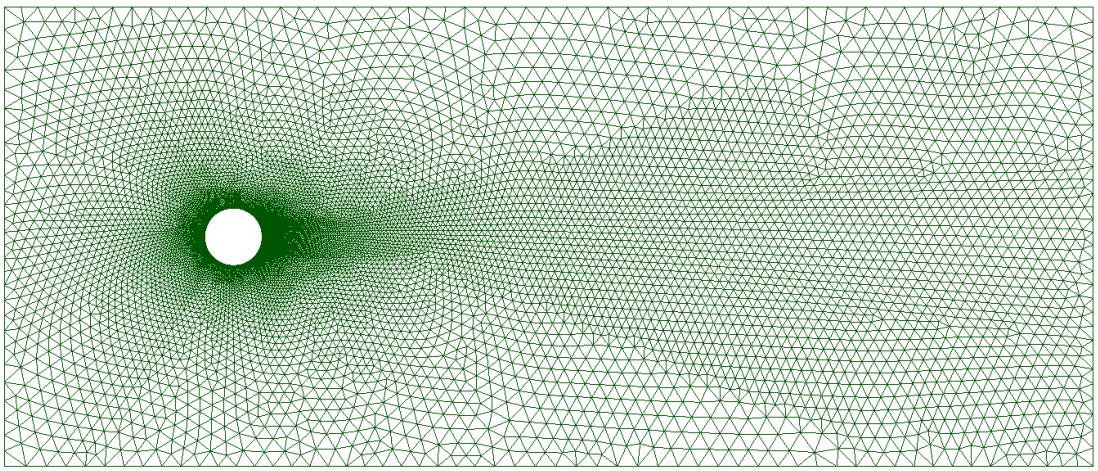


Figure 4.29: Mesh used for the flow behind a cylinder

The objective of this subsection is to observe the frequency and amplitude errors compared to the monolithic solution for the different pressure segregation schemes. Both BDF2 and BDF3 time integration scheme are used . The velocity and pressure contours obtained with the velocity correction BDF2 fractional step scheme with continuous Laplacian are shown in Figure 4.30. Note that the pressures in the outlet are not prescribed to zero as mentioned in the section on open boundary conditions.

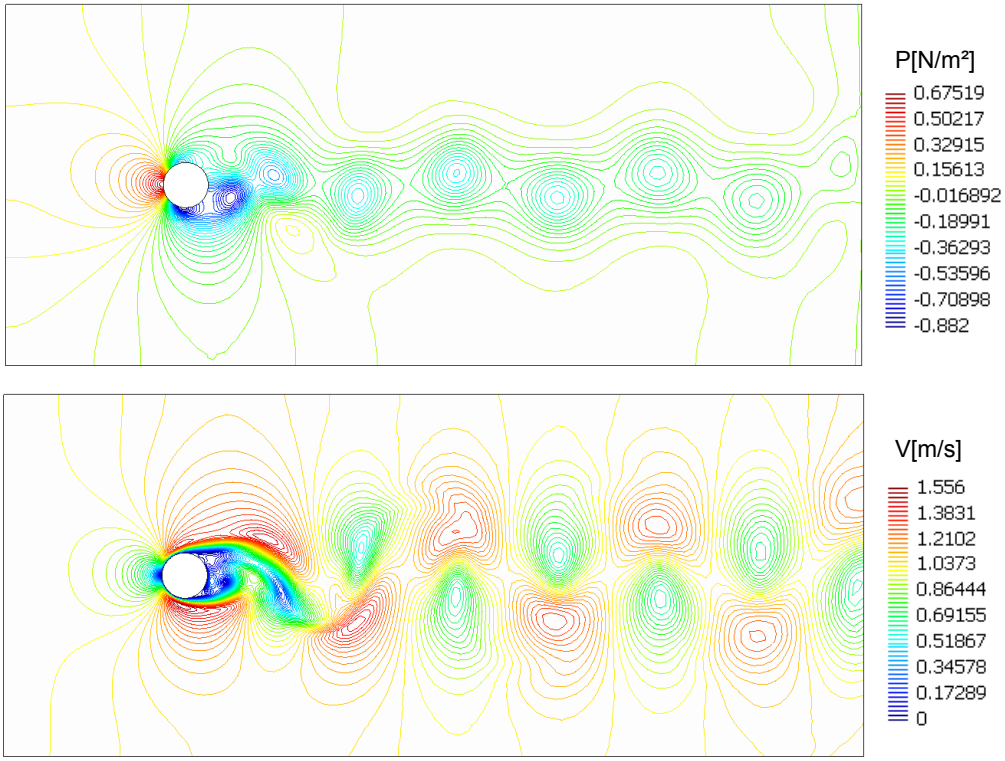


Figure 4.30: Pressure (top) and velocity (bottom) at $t = 100s$

VC - PC comparison using the fractional step scheme with both Laplacians

Figures 4.31 and 4.32 show the Lift and Drag coefficients for the fully developed flow using VC and PC BDF2 fractional step schemes with both Laplacians. For the Lift coefficient, all of the schemes give pretty accurate results. Therefore we shall use the Drag coefficient to compare the different schemes.

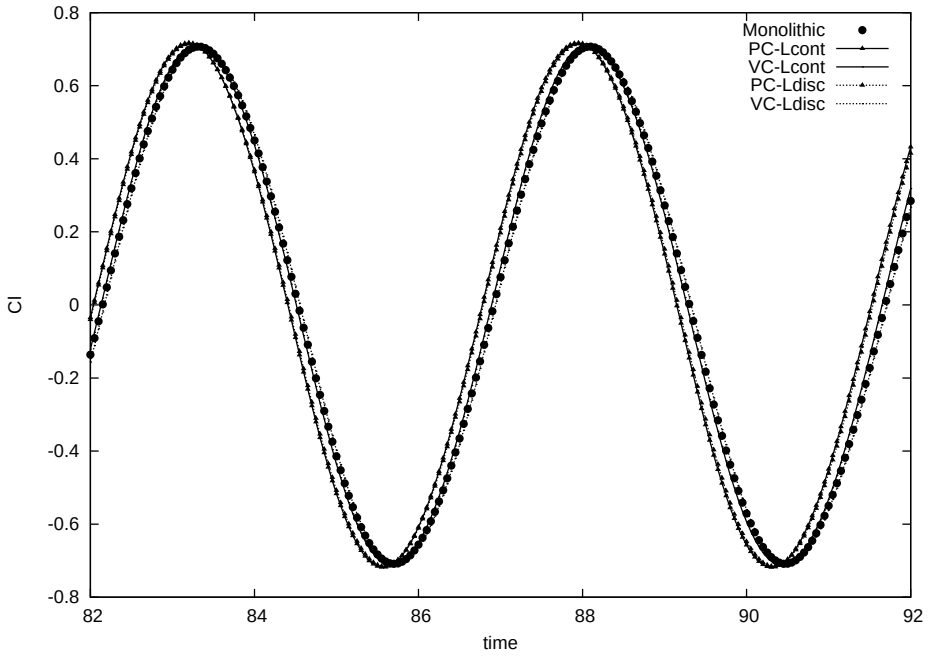


Figure 4.31: Lift coeficient

The first observation is that the VC scheme provides much better results than the PC scheme both with the discrete and continuous Laplacians. The results obtained with the discrete Laplacian are better than those obtained with the continuous Laplacian as one could expect but even the VC scheme with continuous Laplacian shows smaller errors than the PC scheme with discrete Laplacian. When the VC scheme with a discrete Laplacian is used the errors are very small, much smaller than those obtained with a continuous Laplacian (which are already quite small). Therefore we can say that when the rest of the errors are sufficiently small, the real advantage of using a discrete Laplacian can be observed.

An explanation for the advantage of the VC scheme is that we are using a second

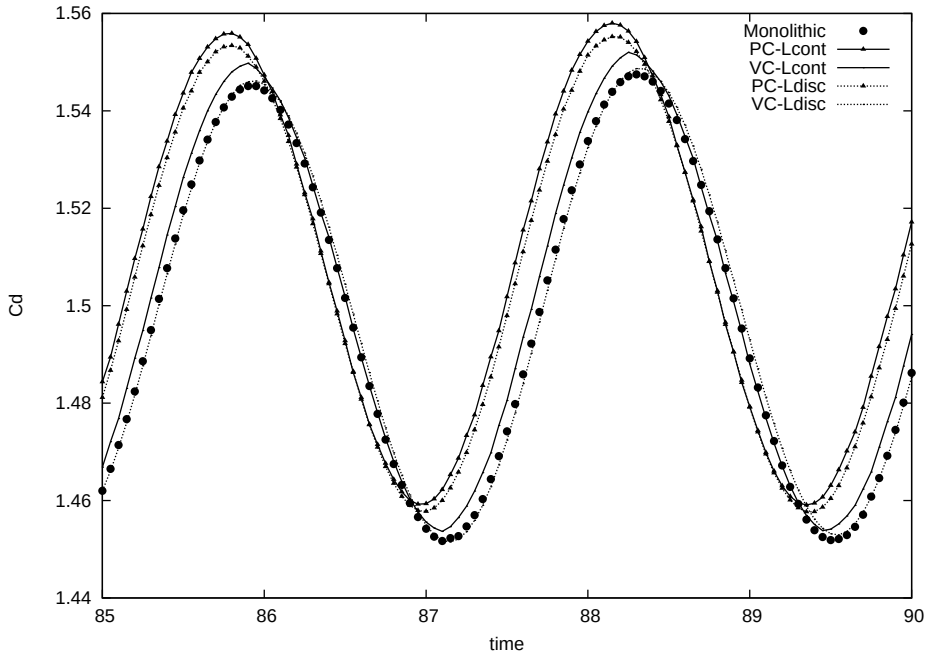


Figure 4.32: Drag coeficient

order extrapolation for the velocity but only a first order extrapolation for the pressure. The reason for doing this is that, as is well known for pressure correction schemes, second order extrapolations for the pressure lead to unstable schemes. We have verified this behavior also in this example. The pressure correction scheme with second order pressure extrapolation diverges after some few steps. We have also tested the VC scheme with continuous Laplacian that uses a second order pressure extrapolation and introduces a third order splitting error to verify that it is unstable as already pointed out in [4]. The VC scheme with continuous Laplacian and first order pressure extrapolation works fine so we can attribute the instability to the use of second order extrapolations for the pressure as in the pressure correction scheme. The excellent results obtained with the VC scheme with discrete Laplacian can be attributed to the fact that it needs no pressure extrapolation and only uses a second order velocity extrapolation leading to a third order splitting error. In the VC scheme with continuous Laplacian the use of a first order extrapolation inhibits the third order accuracy as will be shown in the convergence example in the next subsection. Finally, since the VC scheme with continuous Laplacian uses a mix of a

second order extrapolation for the velocity and a first order extrapolation for the pressure it seems reasonable that it introduces less error that the PC scheme that only uses a first order pressure extrapolation independently of which Laplacian is used.

In order to verify that the advantage of the VC scheme comes from the use of a second order extrapolation we have tested it with an first order extrapolation for the velocity. In Figure 4.33 the results obtained with such VC scheme with Discrete Laplacian are compared to the results already shown in Figure 4.32 for the monolithic and PC schemes. Using a first order extrapolation for the velocity in the VC scheme the results have a similar accuracy to the ones obtained with the PC scheme and a much lower accuracy than the ones obtained with a second order extrapolation for the velocity. Therefore we can conclude that the advantage of the VC scheme comes from the fact that it only needs a second order extrapolation when the discrete Laplacian is used. It introduces a third order splitting error.

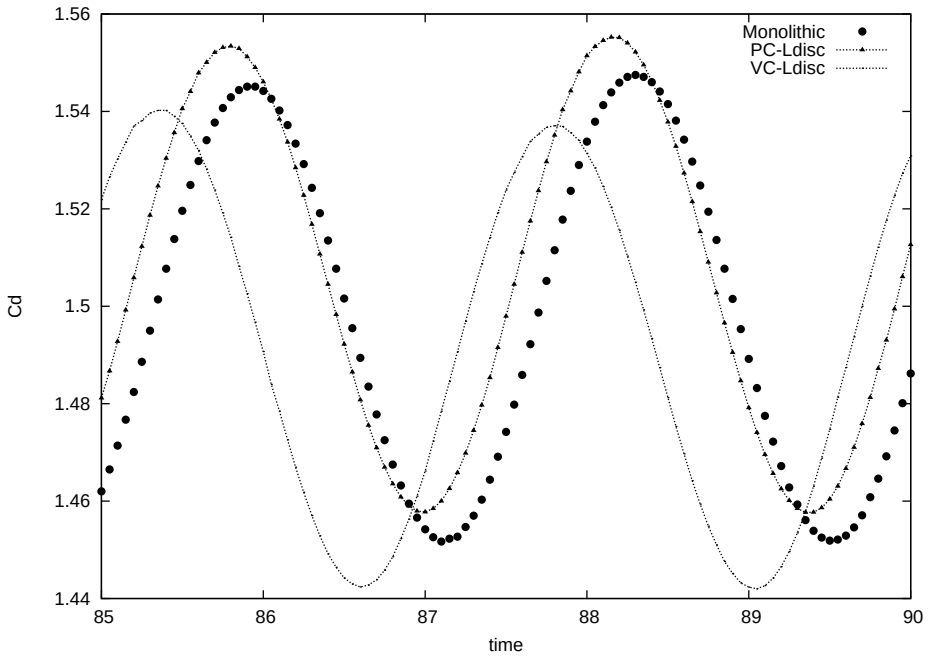


Figure 4.33: Drag coeficient using a first order extrapolation for the VC scheme

In Table 4.2 we compare the total CPU time for the different fractional step versions. The VC schemes turn out to be slightly slower ($\approx 5\%$) than their PC counterparts but

	PC	VC
L cont	$66979s$	$69642s$
Ldisc	$80274s$	$83020s$

Table 4.2: Total cpu time for the fractional step schemes

they provide more accurate results. The use of the continuous Laplacian provides smaller ($\approx 15\%$) CPU times than the discrete Laplacian but also less accurate results. The use of the VC scheme with discrete Laplacian is the slowest option but it provides results that are nearly identical to the monolithic ones.

Since we have obtained a VC scheme with a third order splitting error it seems reasonable to combine it with a third order time discretization such as BDF3. The results obtained with the BDF3 VC fractional step scheme for the drag coefficient where practically identical to the ones obtained with the BDF2 scheme. In order to see the enhancement obtained with the BDF3 time discretization a more sensitive value than the drag coefficient needs to be used. In Figure 4.34 the horizontal velocity at a node located at $(9.759,\ -0.0426,\ 0.2)$ is presented using both the monolithic solver and the fractional step VC scheme with discrete Laplacian. It can be observed that the errors introduced by switching from a third order time discretization to a second order one are more important than the errors introduced by the splitting. We have verified that if a smaller time step is used the results obtained with both the BDF2 and BDF3 become closer. They are very similar to the ones obtained with the BDF3 scheme and the original time step showing the advantage of using a third order scheme.

VC - PC comparison using the predictor corrector scheme with both Laplacians

In the case of the predictor corrector (Imon) versions the comparison of the Lift and Drag coefficients provide little or no information since all of them converge to the same result.

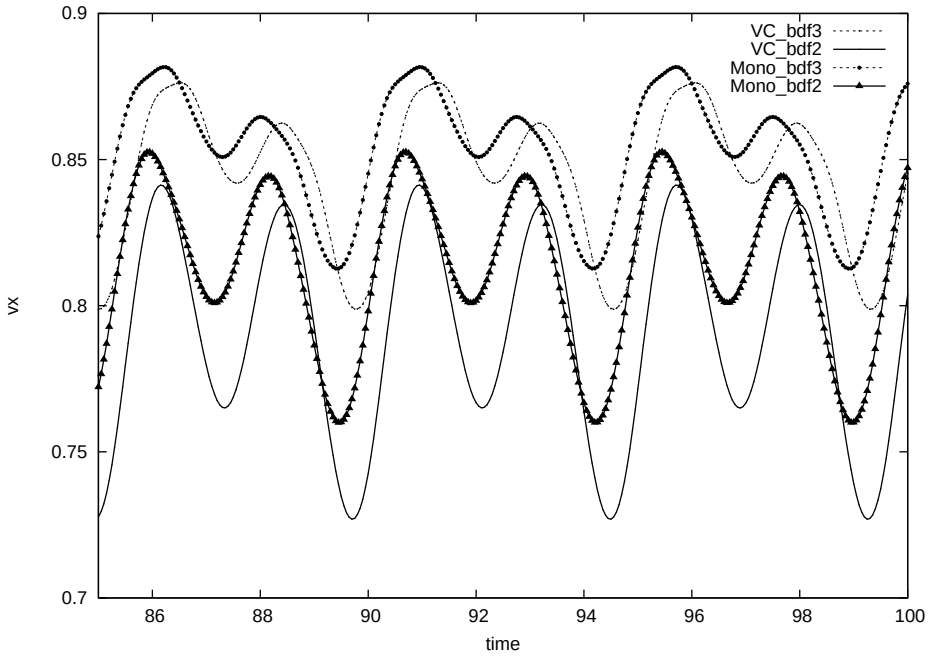


Figure 4.34: Horizontal velocity at a node behind the cylinder

In this case we will be interested in the number of iterations needed to converge to such result. The lower the number of iterations the better the method. When the flow is fully developed the number of iterations per time step remains constant. The convergence tolerance we have used is 10^{-5} based on the variation of the L2 norm of the velocity divided by the norm of the velocity. The number of iterations per time step for each of the schemes are shown in Table 4.3

	PC	VC
L cont	9	7
Ldisc	7	6

Table 4.3: Number of iterations per time step

We can see that the results for the number of iterations agree with what we have observed for the fractional step case for the accuracy of the different options. The VC scheme with discrete Laplacian produces the best results and the PC scheme with

continuous Laplacian the worse ones. Another important parameter to take into account when comparing different schemes is the computational efficiency. In Table 4.4 we compare the total CPU time for the different schemes. The VC schemes turn out to be more efficient ($\approx 20\%$) than their PC counterparts. On the other hand, when converging to the monolithic solution, the use of the continuous Laplacian provides smaller ($\approx 15\%$) CPU times.

	PC	VC
L cont	$290891s$	$241132s$
Ldisc	$336465s$	$288588s$

Table 4.4: Total cpu time for the predictor corrector schemes

Richardson iteration for the discrete Laplacian

As in the cavity case, it is interesting to compare the straightforward use of the conjugate gradient algorithm with the use a Richardson iteration preconditioned with the continuous Laplacian. The cylinder case is a more interesting example than the cavity because we are dealing with a 3-D mesh. We compare the total CPU time when both VC and PC fractional step schemes are used in Table 4.5.

	PC	VC
Straight forward CG	$80274s$	$83020s$
Richardson iteration	$80595s$	$82375s$

Table 4.5: Total cpu time with different options for solving the discrete Laplacian

It can be seen that there is no significant difference between using either of the two options. Both of them could be considered as valid and further tests should be performed to select the most efficient option. Taking into account implementation ease, Richardson

iteration might be an attractive option because it can be very simple to code in a program that uses the continuous Laplacian.

4.5.3 Convergence test

The third example is used to test the time convergence rate numerically. It has been borrowed from [52].

The Stokes problem is solved on the unit square, $]0,1[^2$. The force term is set so that exact solution is

$$p(x,y,t) = \cos(\pi x)\ \sin(\pi y)\ \sin(t)$$

$$u(x,y,t) = \pi \sin(2\pi y)\ \sin^2(\pi x)\ \sin(t)$$

$$v(x,y,t) = -\pi \sin(2\pi x)\ \sin^2(\pi y)\ \sin(t).$$

The domain is discretized using $Q2/Q2$ finite elements of size $h = 1/40$. Boundary and initial conditions are forced to satisfy the previous equations. The time step size we use varies from $0.0025s$ to $0.1s$ and the results at $t = 1.0s$ are presented.

In Figure 4.35 we present the convergence results for the L2 norm of the velocity error using a monolithic formulation. This results can be used as a reference against which the results obtained with the fractional step results can be compared because they have no splitting error. It can be observed that the monolithic BDF1 scheme shows the correct order of convergence. For the third order scheme the error due to the temporal discretization is smaller than the error due to the spatial discretization. For the second order scheme the error due to the temporal discretization is only noticeable for the bigger time steps.

In Figure 4.36 the results obtained with the pressure correction fractional step scheme are presented. The discrete Laplacian is used and some interesting observations on what happens when the continuous Laplacian is used shall be postponed until the end of the subsection. The results with BDF3 are not included because as we are using a first order pressure extrapolation the results cannot be third order accurate. Both schemes show the correct order until the spatial discretization error becomes dominant. Comparing with

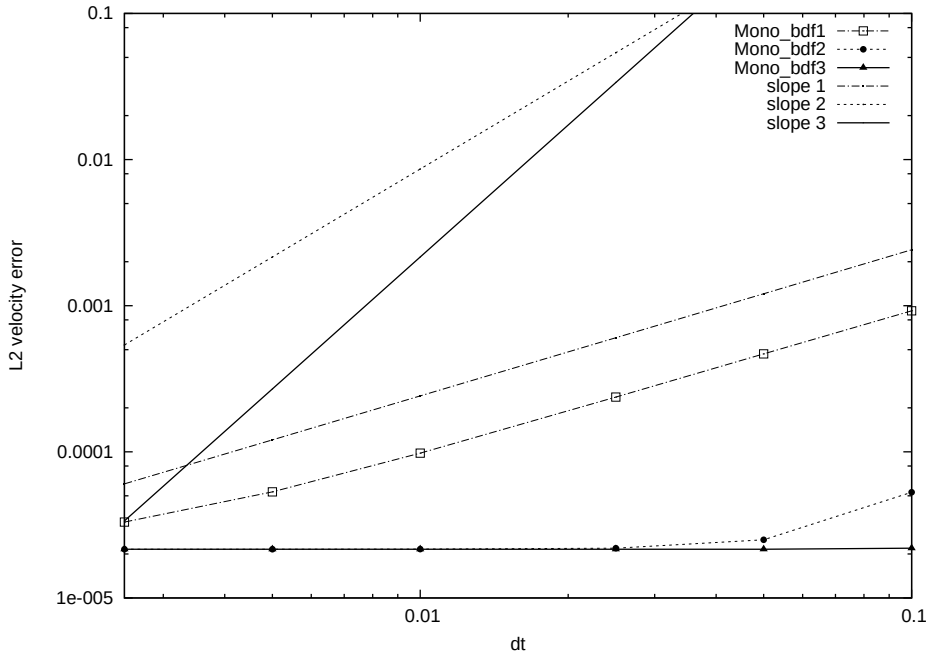


Figure 4.35: Convergence test with monolithic solver

the monolithic example one can see that in the BDF2 case the errors due to the time discretization are much smaller than those due to the splitting. Therefore this is a good example to test a pressure segregation scheme.

Finally we present the results with the velocity correction fractional step scheme where we expect to obtain third order accuracy (Figure 4.37). Both BDF2 and BDF3 results show third order convergence. In the BDF2 case this can be explained by the fact that the error due to the temporal discretization is small compared with the splitting error which is third order accurate. The spatial error is important and it limits the range where third order accuracy can be observed. Therefore the results have been repeated on a mesh with size $h = 1/200$.

In Figure 4.38 the convergence results using a monolithic formulation on the fine mesh are presented. As one would expect the error due to the spatial discretization is reduced 5^3 times thanks to the use of $Q2/Q2$ finite elements. With the BDF2 scheme the second order slope can easily be observed. For the third order scheme the spatial discretization error soon becomes dominant and the third order slope can only be seen for the two bigger

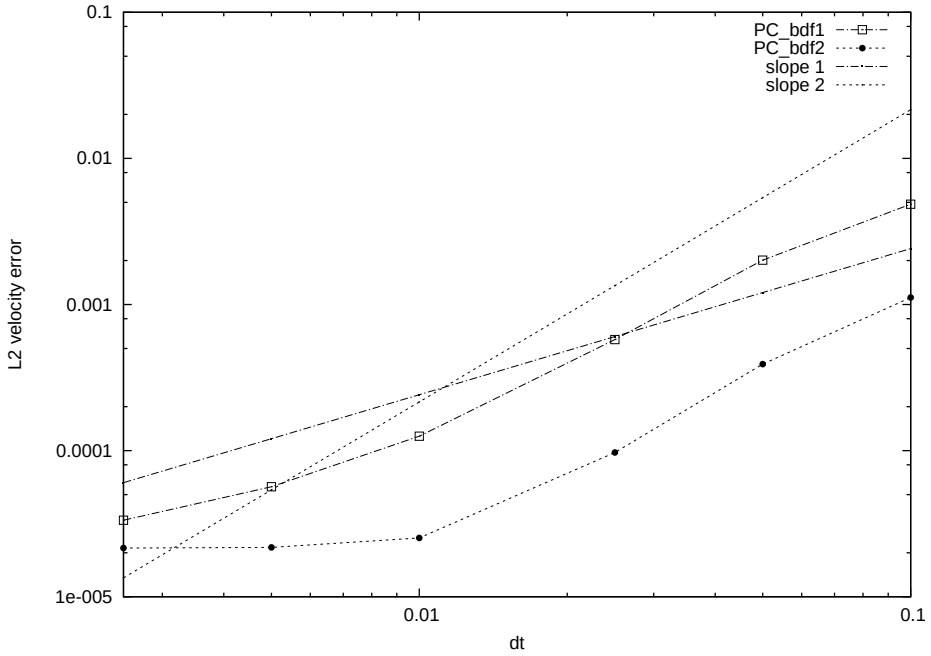


Figure 4.36: Convergence test with pressure correction scheme

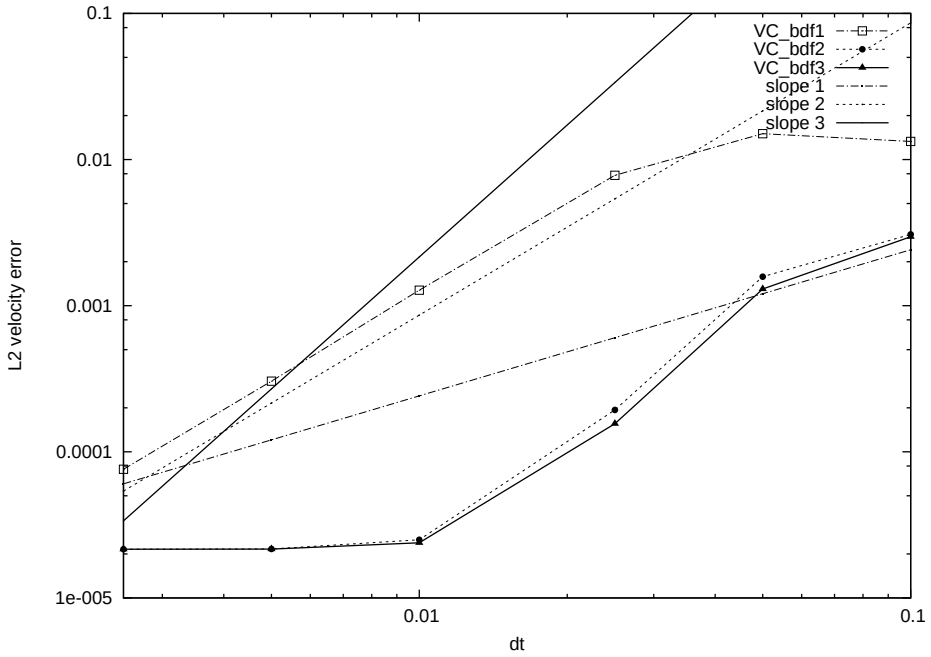


Figure 4.37: Convergence test with velocity correction scheme

time steps.

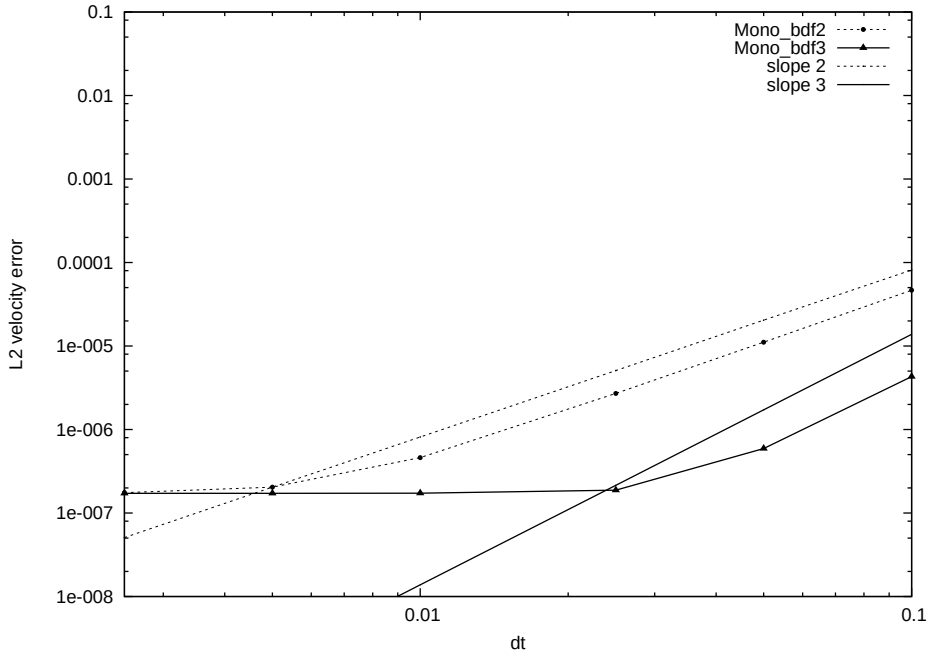


Figure 4.38: Convergence test with monolithic solver on fine mesh

For the velocity correction scheme both BDF2 and BDF3 schemes show third order accuracy because as we have already mentioned for the coarse mesh the splitting error is more important than the time discretization error (Figure 4.39).

Finally we present some results with the VC BDF3 scheme and continuous Laplacian (Figure 4.40). The third order is lost and only second order accuracy is obtained due to the error introduced by the approximation of the discrete Laplacian by the continuous one. Remember we are using a first order extrapolation for the pressure because we have seen that a second order extrapolation leads to an unstable scheme. We have also included in the comparison the results obtained with the discrete Laplacian solved by a Richardson iteration. Actually only three Richardson iterations have been allowed per time step. It is interesting to note how two extra Richardson iterations allow to recover results that are nearly third order accurate and very close to the ones obtained when the discrete Laplacian is solved with a conjugate gradient method.

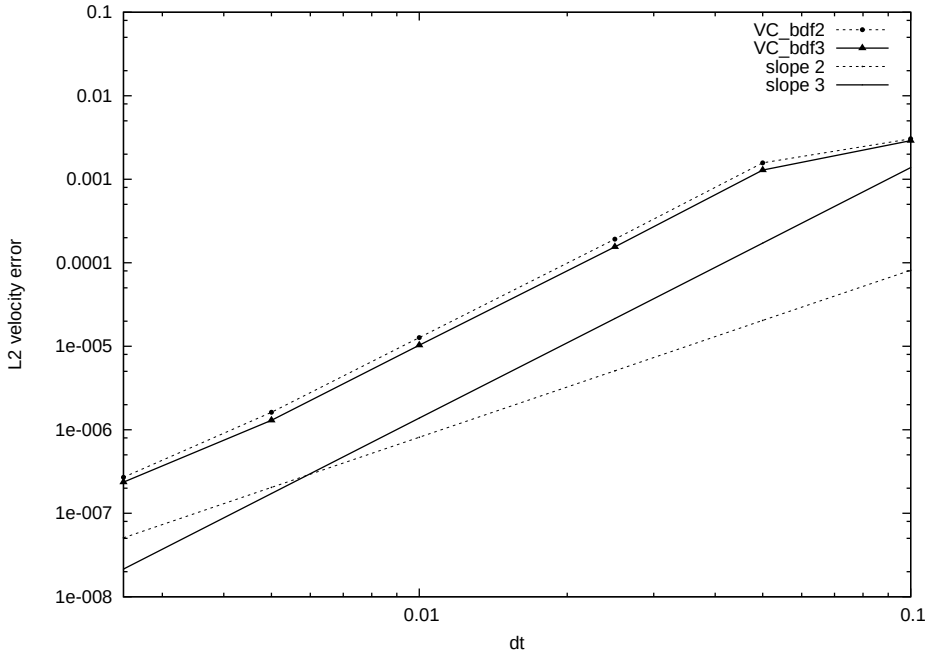


Figure 4.39: Convergence test with velocity correction scheme on fine mesh

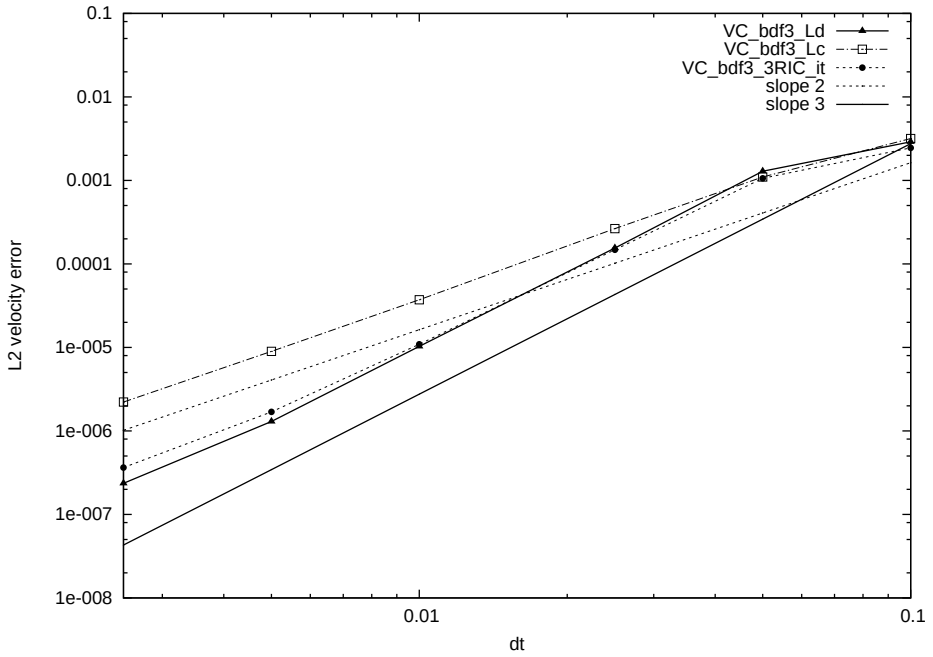


Figure 4.40: Convergence test with BDF3 VC scheme and different Laplacians

4.5.4 Results with the rotational form

In this subsection we present some results using the rotational form of the pressure correction fractional step scheme in the previous three examples. The objective is to verify the correct implementation in the rotational version in the pressure stabilized case and gain some idea of the advantages of the rotational form.

First the driven cavity example is analyzed. In Figure 4.41 we show the transient residual evolution using the PC BDF1 fractional step scheme with standard and rotational versions. Both the continuous and discrete Laplacians have been used. It can be observed that better results are obtained with the rotational version independently of which Laplacian is used.

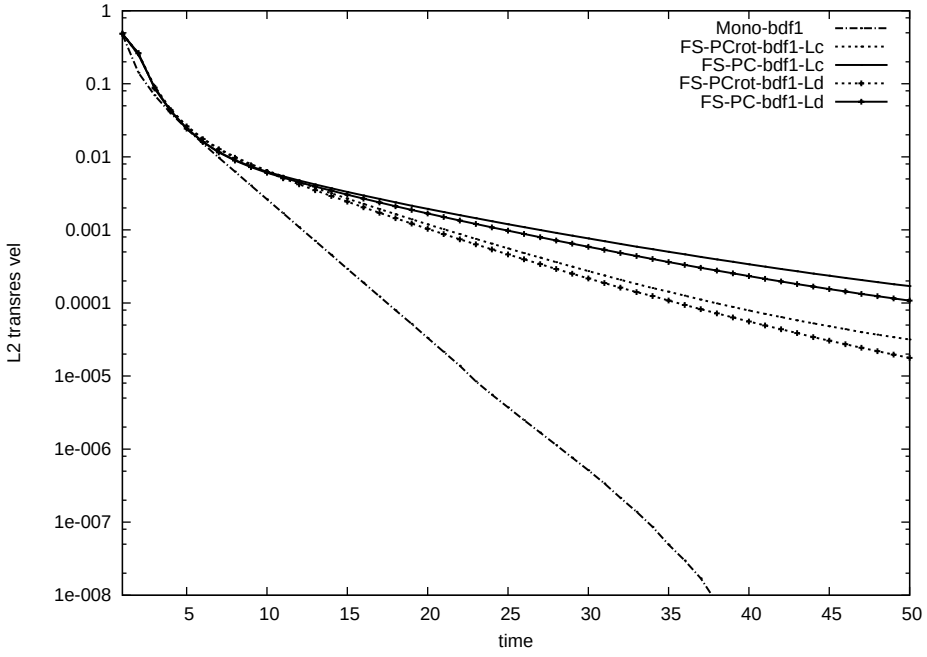


Figure 4.41: Transient residual BDF1 - Influence of the rotational form using both Laplacians

In Figures 4.42 and 4.43 the velocity and pressure errors are presented. Again the rotational version provides improved results independently of which Laplacian is used. Using the BDF2 scheme the advantage of the rotational version in the evolution of the transient residual and the pressure and velocity errors has also been confirmed.

For the second example, the flow behind a cylinder, the drag coefficient is used to

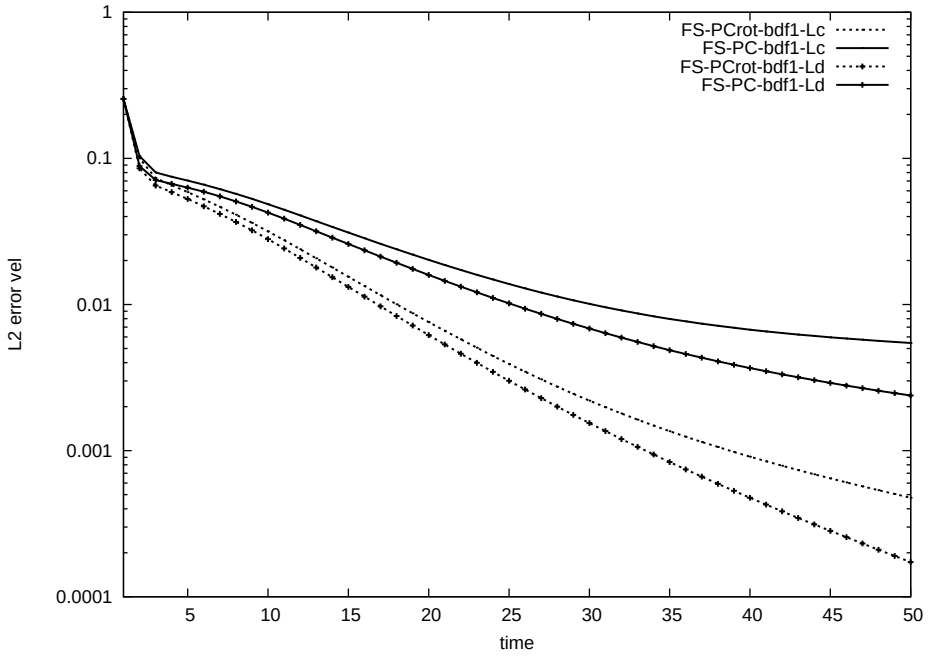


Figure 4.42: Velocity error BDF1 - Influence of the rotational form using both Laplacians

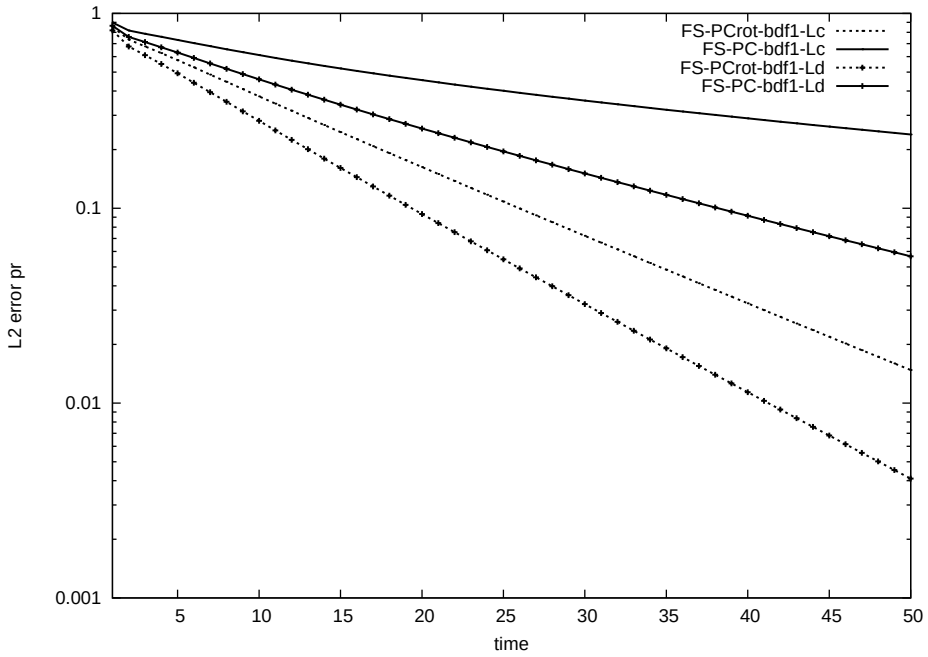


Figure 4.43: Pressure error BDF1 - Influence of the rotational form using both Laplacians

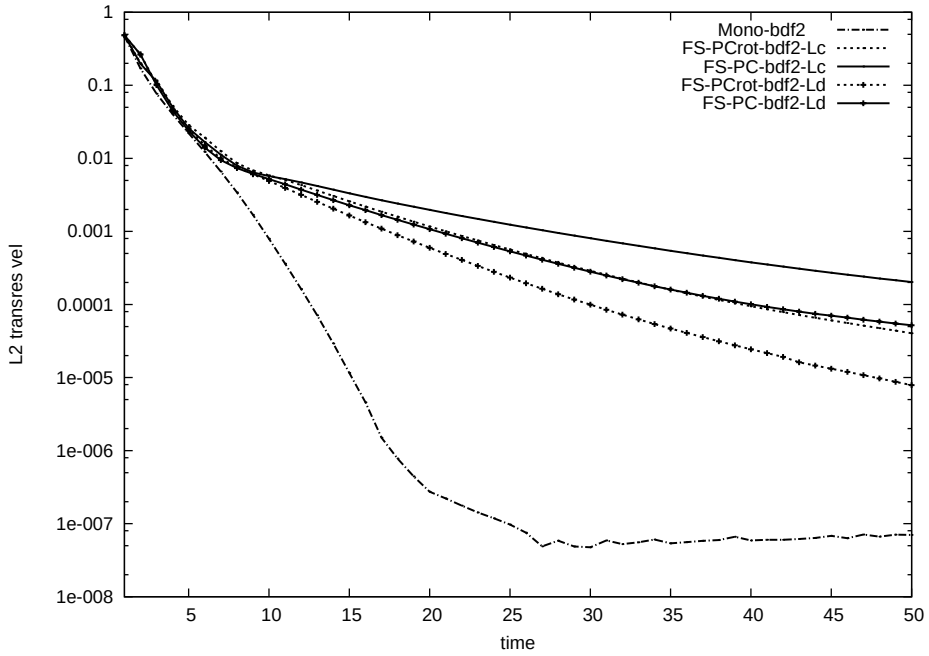


Figure 4.44: Transient residual BDF2 - Influence of the rotational form using both Laplacians

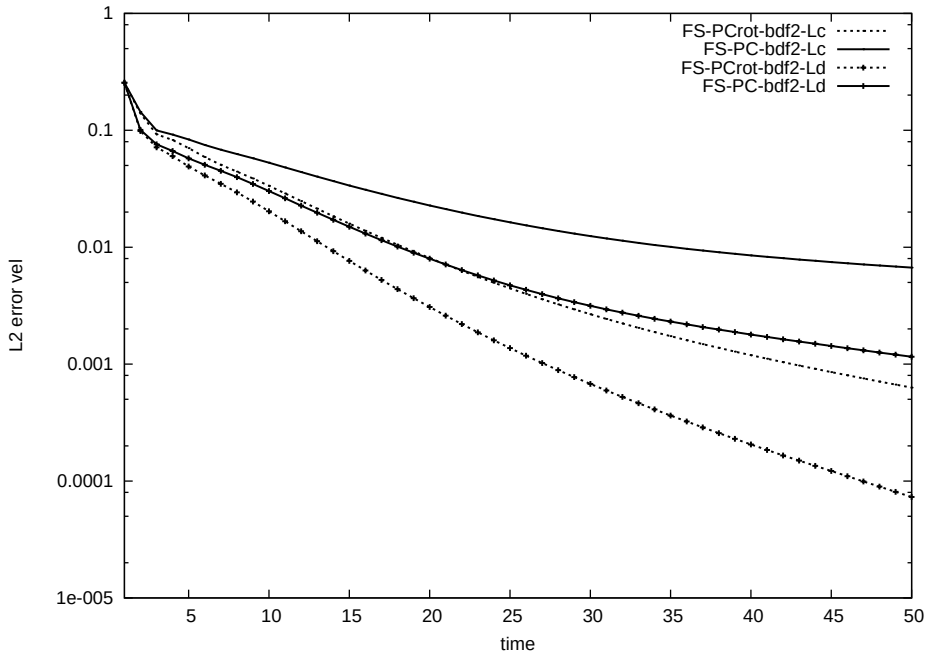


Figure 4.45: Velocity error BDF2 - Influence of the rotational form using both Laplacians

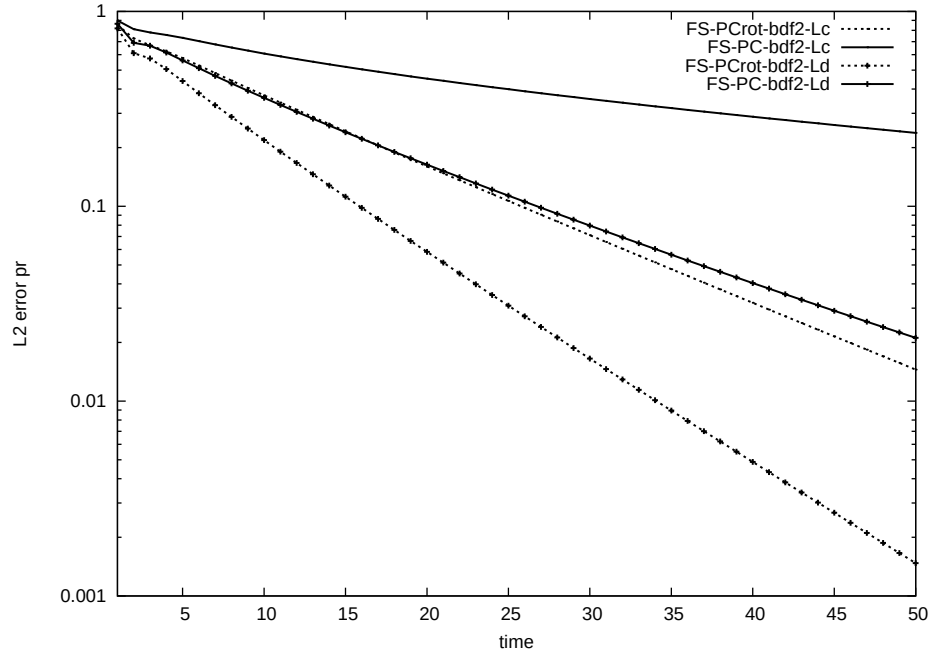


Figure 4.46: Pressure error BDF2 - Influence of the rotational form using both Laplacians

evaluate the advantages introduced by the rotational form. The BDF2 scheme with both Laplacians has been tested. Contrary to what happens in the previous example, for the cylinder the drag obtained with the rotational version is indistinguishable from the one obtained with the standard version (Figure 4.47). We believe that this can be attributed to the fact that in this example the viscous forces are less important.

For the example used in the convergence test the errors in the pressure at $t = 1.0s$ for a $0.01s$ time step are used to compare the standard and rotational versions. Both the continuous and discrete Laplacians have been used. When the standard version is used the errors concentrate close to the upper and lower boundaries where the exact pressure has a non zero normal gradient. This can be explained by the fact that the non rotational pressure correction fractional step enforces a Non-physical boundary condition $\partial_n p^{n+1} = \partial_n p^n$ [50] at the Dirichlet boundaries. We have added the results with the discrete Laplacian to show the same behavior is observed irrespective of which Laplacian is used. When the rotational version is used the error is smaller and concentrates only in the corners. This has also been observed in [50] where it has been conjectured that it is

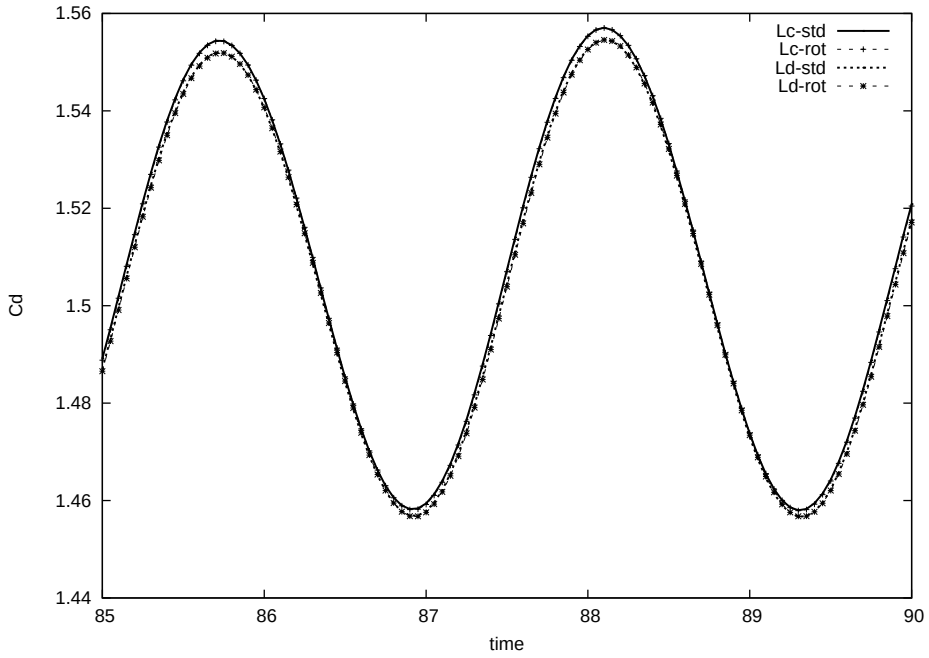


Figure 4.47: Drag coefficient rotational and standard form with both Laplacians

due to the lack of smoothness in the domain.

4.6 Conclusions

In this Chapter we have presented pressure correction and velocity correction pressure segregation schemes. We have implemented them in our code and tested them to gain some experience in their comparative behavior. Both the discrete Laplacian and the (more usual) approximation by the continuous one have been implemented. For the solution of the discrete Laplacian two options have been implemented. The first one is the straightforward application of a conjugate gradient iterative procedure. The second option is to use a preconditioned Richardson iteration with the continuous Laplacian as preconditioner.

The rotational version of the pressure correction scheme has also been implemented. Some particularities arise when a pressure stabilized scheme is used. Up to the moment it had only been used with elements that satisfy the Inf-Sup condition and do not require

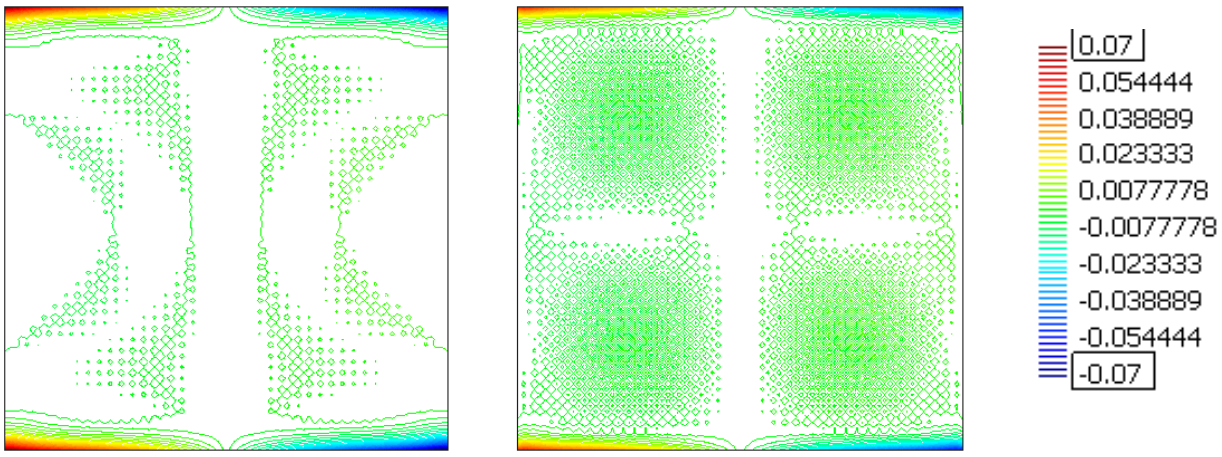


Figure 4.48: Pressure error for the non rotational form with continuous (left) and discrete (right) Laplacians

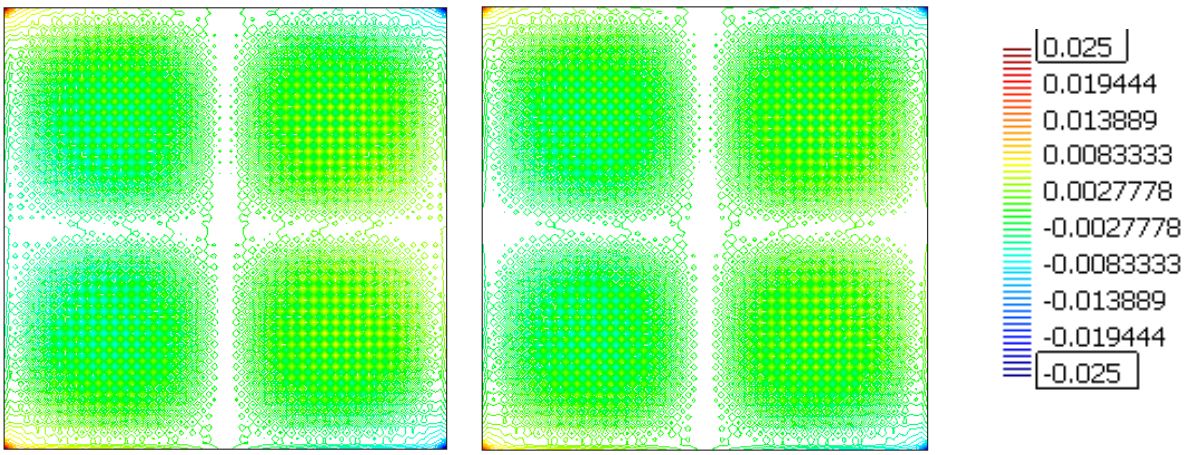


Figure 4.49: Pressure error for the rotational form with continuous (left) and discrete (right) Laplacians

pressure stabilization. Moreover it has been shown numerically that the advantages introduced by the rotational version are present when either the continuous or discrete Laplacians are used.

Three numerical example have been used. The first one is the evolution of the flow in a driven cavity to the steady state. It has already been used in [4] where only results with the continuous Laplacian have been presented. In this thesis the effect of using a discrete Laplacian has also been taken into account. As in [4] the evolution of the transient residual is used to compare the two methods. In this thesis the errors obtained by comparing the velocity and the pressure with the monolithic solution are also presented. The pressure and velocity correction schemes show a similar behavior with some advantage for the velocity correction version. This agrees with what has been observed in [4]. With the discrete Laplacian a similar behavior is observed but most of the results are improved. The enhancement due to the use of the discrete Laplacian is most noticeable in the pressure errors. In the predictor corrector case, the number of iterations needed to converge to the monolithic solution also show some improvement when the velocity correction scheme is used. For this example the preconditioned Richardson iteration used to solve for the discrete Laplacian proves more costly than the straightforward use of conjugate gradient method. It is interesting to note that no instabilities have been observed despite we have used a second order extrapolation for the velocity that leads to a third order splitting error when the velocity correction BDF2 scheme with discrete Laplacian is used.

For the cylinder example the absence of stability problems due to the second order velocity extrapolation is once again verified. The correct behavior in this example is particularly important because it is a complicated oscillatory transient flow typically used as a benchmark for transient flow algorithms. The real advantage of using a second order extrapolation for the velocity becomes clearly noticeable in this example. When used with a discrete Laplacian in the velocity correction scheme it leads to a third order splitting error evidenced in the clear superiority observed in the prediction of the drag coefficient. The use of the continuous Laplacian diminishes the advantage of the VC scheme despite it is still superior to the pressure correction scheme. The reason is that when a continuous

Laplacian approximation is used a pressure extrapolation is also needed. If a second order extrapolation for the pressure is used to obtain a third order splitting error we have observed that the method becomes unstable. This agrees with the well known behavior for pressure correction schemes where second order pressure extrapolations make the scheme unstable. We believe that the impossibility of obtaining stable third order results in [4] can be attributed to the use of the continuous Laplacian approximation with second order pressure extrapolation and that different conclusion could have been obtained if the discrete Laplacian had been tested.

Since the VC scheme with discrete Laplacian provides a third order splitting error it has also been combined with a BDF3 time discretization to obtain a third order temporal error. The enhancement introduced by the BDF3 time discretization is not noticeable in the drag coefficient and a more sensitive parameter such as the horizontal velocity in some node behind the cylinder has been used to show the advantage introduced by the third order time discretization.

Regarding the use of the discrete Laplacian in the VC case we have already mentioned that it provides much better results because it allows to obtain a third order splitting error. In the pressure correction case it also introduces some advantage but it is much less significant than in the velocity correction case. Despite the use of the discrete Laplacian leads to a an increase in the computational cost of 15% the advantages observed in the velocity correction case easily justify this cost.

In the predictor corrector case the velocity correction scheme also shows advantage over the pressure correction scheme. It leads to fewer predictor corrector iterations per time step and consequently a reduced computational cost of approximately 20% . In the predictor corrector case the use of the continuous Laplacian results in an increase of CPU times of approximately 15%. Regarding the use of the preconditioned Richardson iteration to solve for the discrete Laplacian in this example it has resulted in approximately the same computational times as the straightforward use of a conjugate gradient solver.

In the convergence test borrowed from [52] the expected convergence slope for the L2 velocity error is verified. For the pressure correction scheme a second order slope

is observed with the BDF2 time discretization. With the first order time discretization second order slope is observed when the splitting error is dominant and first order slope is observed when the temporal discretization error is dominant. For the velocity correction case it is observed that both BDF2 and BDF3 schemes show third order accuracy. For the second order time discretization this can be explained by the fact that the splitting error which is third order accurate is dominant. When the continuous Laplacian approximation is used the third order accuracy obtained with the velocity correction scheme is lost. This is caused by the use of a first order pressure extrapolation. It has also been shown that the use of a preconditioned Richardson iteration with only three iterations allows to recover results that are nearly third order accurate and very close to the ones obtained with the discrete Laplacian.

Regarding the use of the rotational form in the pressure correction scheme, improvements have been observed in the first and third examples. It is interesting to note that this improvements have been observed irrespective of whether the continuous or discrete Laplacian is used. Moreover we understand that this is the first time the rotation form has been used with a pressure stabilized scheme. In the cylinder example the use of the rotational form has introduced no improvement. The rotational form to the velocity correction pressure stabilized scheme can be an interesting extension for future developments. The results on the advantages of the rotational version for real applications are not conclusive for the moment. Following [15] we can say that their importance grows in applications in which the stresses or other pressure dependent quantities must be computed at solid walls. Moreover since pressure segregation methods are quite often applied to problems in which the viscosity is small, the improvement introduced by the rotational form in such cases can be negligible. This is what we believe happens in the cylinder case.

From the previous examples we can conclude that the superior behavior of velocity correction methods can be attributed to the fact that second order velocity extrapolations lead to stable schemes. Instead second order pressure extrapolations lead to unstable schemes. This not only happens in the pressure correction case but also in the velocity

correction with continuous Laplacian approximation. Therefore we can say that the use of a discrete Laplacian is more significant in the velocity correction scheme because it provides a 'pure' VC scheme where no pressure extrapolation is needed. The use of the Richardson iteration with a limited number of iterations as in the convergence test can be a cost effective option in this direction.

Regarding the extension to interface problems, the velocity also seems better to extrapolate than the pressure. For example, in the simple 3D vertical tube presented in Chapter 2 the velocity extrapolation would be exact while the pressure extrapolation would be quite poor specially close to the interface. Despite this example is ideal and very simple it can be quite representative of what happens during mould filling simulations. For the enriched pressure model presented in Chapter 2 the use of a velocity extrapolation makes the method much simpler than a pressure extrapolation because the pressure is enriched and the velocity is not. If a pressure extrapolation were used, when solving for time step $n+1$ one would need the pressure extrapolation $\widetilde{p}^{n+1} = p^n$ that includes an enriched component that depends on the position of the interface at time n. This does not seem appealing. For the previous reasons for interface problems we have chosen to implement the velocity correction scheme.

Chapter 5

Mould Filling

5.1 Introduction

In this chapter the utility of the methods presented in previous chapters is explored in the context of mould filling applications. Examples borrowed directly from the foundry are used to test the improvements introduced by the proposed methods.

The numerical simulation of mould filling processes has become a widespread tool for improving casting technology. Regions with high velocities that can lead to premature wear of the mould can be predicted. The quality of the resulting piece can also be improved, for example, by determining regions of possible air entrapment. An overview of computational methods for free surface flows in casting and Industry-Standard Mold-Filling codes can be found in [40].

Contrary to what one might intuitively think, we have observed that in mould filling problems, lower filling velocities typically lead to more complex simulations. That is to say, low Froude number flows pose special difficulties for two phase flows. The lower the Froude number, the higher the importance of the gravitational forces. Since the spatial distribution of the gravitational forces is determined by the position of the interface, the coupling between the position of the interface and the resulting flow increases as the Froude number decreases. An accurate representation of the pressure in the elements cut

by the interface is needed for such flows. By enriching the pressure finite element shape functions or by using a free surface model we have obtained important improvements in simple examples. In this work we extend the application of both models to real mould filling problems.

In Chapter 2 an enriched pressure interpolation for two phase flows that provides significant improvement over the usual two phase flow model in low Froude number simulations has been presented. In Chapter 3 a free surface model on a fixed mesh that only simulates the region occupied by the fluid and neglects the influence of air has been presented. This method has also shown good results for low Froude number flows thanks to a careful treatment of the elements cut by the front that allows to accurately impose the boundary conditions at the interface. Moreover in Chapter 4 pressure segregation methods that allow to uncouple the solution of the velocity and the pressure have been presented. For interface problems velocity correction methods have been implemented. Therefore we are left with a total of four options to solve for mould filling interface flows. The two models for interface flows, the enriched pressure two phase model and the free surface model, are combined with two solution strategies, the monolithic solver and the velocity correction solver. As we have mentioned in the conclusions of the previous Chapter, we have preferred the velocity correction scheme instead of the pressure correction, not only because it provides better results in the one fluid case, but also because the velocity seems easier to extrapolate than the pressure in interface problems.

For the velocity correction scheme only the discrete Laplacian will be used. The reason for doing this, despite it can be computationally more expensive, is that it implies one approximation less. In the free surface case the results obtained with the velocity correction scheme are for the moment not very satisfactory. As we shall show in Section 5.3, the convergence has been complicated and the computational time has risen significantly. Therefore it does not seem wise for the moment to add an additional source of error. In the enriched pressure two phase model the velocity correction scheme provides results that are comparable with the ones obtained with the monolithic solver. When the enriched pressure is used the pressure is discontinuous on cut element faces and therefore

we prefer to postpone the exploration of the use of the continuous Laplacian until the enriched pressure velocity correction scheme seems a more interesting option to use. For the moment the free surface model with a monolithic scheme seems to be the best option as we shall show in Section 5.2.

This Chapter will be organized as follows. In Section 5.2 results with the free surface model used with a monolithic solver are presented. The application of the free surface model with a velocity correction scheme is presented in Section 5.3. Sections 5.4 and 5.5 deal with the application of the enriched pressure two phase flow model with monolithic and velocity corrections schemes respectively.

Three mechanical pieces will be used to test the different methods. The first one is a hollow mechanical piece made of steel with physical properties: $\rho = 7266.0$ and $\mu = 6.7 \times 10^{-3}$ (SI units). This piece is interesting because it has relatively thin walls which make the mesh quite complex. The code is forced to obtain acceptable results with few elements in the thickness. The arrangement we simulate consists of two pieces together with the filling channel used during the actual filling process. The inlet velocity is 0.113 m/s and the size of each piece is approximately $0.16 \times 0.16 \times 0.13$ m^3 . The whole filling process takes 11 seconds.

Two unstructured triangular meshes have been used. The coarse one has 72032 elements and 16149 nodes and the fine one has 575803 elements and 116214 nodes. They are shown in Figure 5.1. The Reynolds number based on the inlet velocity and the length of the filling channel is $Re = 2.45 \times 10^4$, and the Froude number is $Fr = 0.0065$.

The second example is an automotive alloy wheel. The flow is created by applying a pressure on the fluid as is done in the actual filling process for this piece. The flow rate is then determined by the resistance exerted on the fluid. We have observed the friction may be high in the vertical tube through which the molten metal is injected. Therefore, for this case, we will simulate the whole filling channel.

The pressure at the inlet varies linearly from $2.21 \times 10^4 N/m^2$ at the beginning of the simulation to $1.17 \times 10^5 N/m^2$ after 4.4 seconds. The physical properties we have used are those of aluminum, $\rho = 2700.0$ and $\mu = 1.3 \times 10^{-3}$ (SI units). The Reynolds number

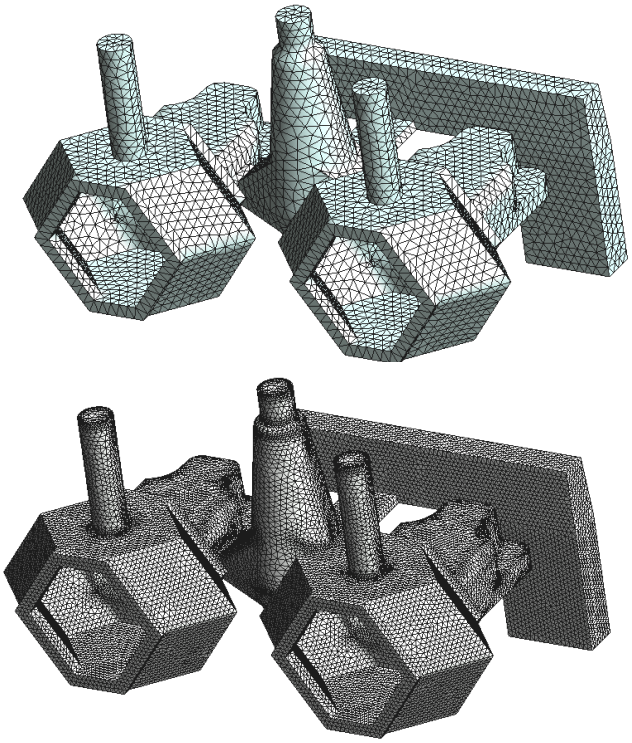


Figure 5.1: Coarse and fine meshes for the hollow mechanical piece

based on a typical velocity inside the wheel (0.5 m/s) and the wheel radius (0.5 m) is $Re = 5.19 \times 10^5$. The Froude number is $Fr = 0.05$. The mesh is formed by 489313 tetrahedral elements and 109318 nodes.

The third piece was presented to us as a really demanding case. It is the shovel for a power shovel. The filling process takes approximately half a minute and the shovel is nearly one meter long. The inlet velocity we have used during the simulation is 0.5 m/s . The Reynolds number based on the previous velocity and length is $Re = 4.44 \times 10^5$, and the Froude number is $Fr = 0.031$. As in the first example the material used is steel. The mesh used for this example consists of 412848 tetrahedral elements and 87010 nodes.

Since the flow in mould filling problems is turbulent the viscosity in the previous equations has been calculated using the Smagorinsky model as $\mu = \mu_L + \mu_T$, where μ_L is the molecular, constant, viscosity and $\mu_T = \mu_T(\boldsymbol{u})$ is the additional turbulent viscosity defined by $\mu_T = C^2\ h^2\sqrt{2\ \varepsilon(u) : \varepsilon(u)}$, where h is the size of the element where it is computed and C^2 is the Smagorinsky constant. The objective of this thesis is not related to the analysis of the influence of the turbulence model and therefore a simple model has been chosen. The Smagorinsky model is also used for mould filling simulations in [16, 46] and in the commercial code Vulcan [118] used to compare against our results in the next Section.

Due to the high Reynolds number of the problems we are dealing with, no slip boundary conditions would require extremely fine meshes along the boundary that would make them computationally unfeasible. The solution we have adopted is to use wall functions [72] that describe the behavior of the flow near a solid wall. The normal component of the velocity is set to zero. In the tangential direction a traction that depends on the velocity at the boundary and is opposed to the direction of the flow is applied:

$$\tau_w = -\rho \frac{u_*^2}{|\boldsymbol{u}|}\ \boldsymbol{u}$$

where u_* can be determined from the following set of equations

$$u^+ = \frac{1}{\kappa} ln\left(1 + \kappa\ y^+\right) + 7.8\left[1 - \mathrm{e}^{-y^+/11.0} - \frac{y^+}{11.0}\mathrm{e}^{-0.33y^+}\right]$$

$$u^+ = \frac{\rho\,|\boldsymbol{u}|\,u_*}{\tau_w},$$

$$y^+ = \frac{\rho\delta u_*}{\mu}.$$

δ is the distance between the computational boundary and the wall, $\kappa = 0.41$ is the Von Karman constant and y^+ and u^+ are non dimensional distances and velocities, respectively.

5.2 Free surface monolithic model

In this Section we present the results obtained with the free surface monolithic model for the previous three pieces. We observe that this model provides the best results of all four analyzed options and therefore this results can be considered as a reference against which the results obtained with the other models can be compared. It has allowed us to use bigger time steps and a lower Smagorinsky parameter than the other methods. Moreover, the convergence of both the nonlinearity and the iterative solver are significatively better than when the enriched pressure two phase flow is used. This leads to significantly improved computational costs. The monolithic scheme does not have splitting errors, that need to be corrected iteratively, as happens when a predictor corrector scheme is used.

Split OSS stabilization has been used for the Navier Stokes equations but some cases have also been run with Non Split OSS and ASGS and no significant difference has been observed. For the convective nonlinearity Picard iteration is used. The tolerance is set to one percent variation in the L2 norm of the velocity and a maximum of 7 iterations are allowed. Typically the nonlinearity converges in less than three iterations. For the solution of the monolithic system a preconditioned GMRES iterative solver [102] is used. The stopping criteria for the solver is that the residual is smaller than 10^{-6} times the right hand side. A maximum of 500 iterations are allowed but the solver usually converges in

less than 30 iterations. The Krylov dimension is set to 50. An ILUT preconditioner with threshold 0.001 and filling 20 is used [102]. For the Level Set equation the convergence of the GMRES solver is very easy even without preconditioner.

In Chapter 3 two alternatives have been proposed for solving the free surface problem; a FM-ALE model and a simplified Eulerian model. The results obtained with both models are very similar. The results obtained with the simplified Eulerian model shall be presented because they are computationally cheaper. At the end of this Section the results obtained with the FM-ALE model are discussed.

5.2.1 Hollow mechanical piece

Figure 5.2 shows the evolution of the interface for four time steps during the filling process when the coarse mesh is used. In the first step the interface is still inside the filling channel. For the second one it has entered both pieces. In the third one the interface reaches the bottom of each piece. As we will comment later, this is one of the most complicated moments in the simulation. In the final figure more than half of each piece has been filled. The evolution of the front is very similar in both pieces. Despite a coarse grid has been used the evolution of the interface is captured quite satisfactorily as one can observe by comparing with the results shown for the fine mesh in Figure 5.3. For both meshes the time steps size is 0.02 seconds. The Smagorinsky model has been used to take into account turbulence and the constant has been set to $C^2 = 0.05$.

Knowing how the interface evolves is important during the mould design as it can be used to change the position of the inlets or alter the filling velocity to improve the quality of the resulting piece. When defects appear, having some insight on the way the flow evolves is of great help to the foundry person because it is very difficult to actually see what is happening inside the mould.

The evolution of the interface using the fine mesh in shown Figure 5.3. The shape of the interface is smoother than the one obtained with the coarse mesh but there is no mayor difference in the way the flow evolves. The most noticeable change is that for each time

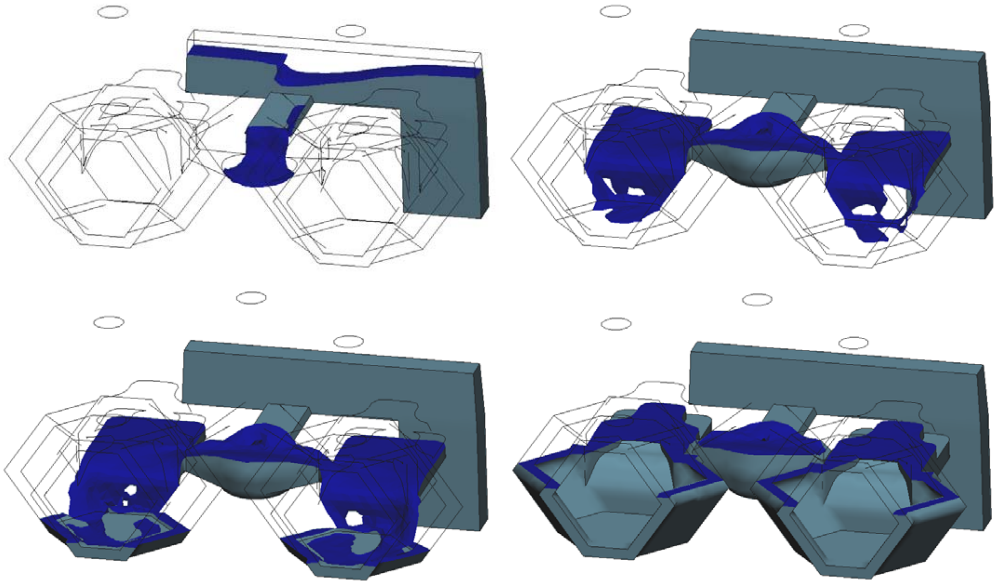


Figure 5.2: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the coarse mesh

step the results obtained with the fine mesh show a bigger percentage of filled volume. This is related to numerical mass losses and is analyzed in more detail in Figure 5.4. Since foundry pieces are usually complex and it is common to fill several pieces at the same time (not only two as in the example) it is important to have a code that can provide the user with acceptable results even with coarse meshes.

In Figure 5.4 we compare temporal evolution of the injected and filled volumes using both meshes. The injected volume is the same for both meshes. The difference between the filled and injected volumes is the numerical mass loss. It is reduced as the mesh is refined as one could expect. The amount of mass loss can give us some idea on the quality of our results and indicate the most complex moments during the simulation. In our example, we can see that the most important mass loss occurs when the filled volume is between 0.0004 m^3 and 0.0006 m^3. It corresponds to the moment when the bottom of each piece is being filled. This suggests that a mesh refinement close to that area might improve the solution. When the fine mesh is used the mass loss is very small. Even with the coarse mesh mass conservation is much better than when the enriched pressure two

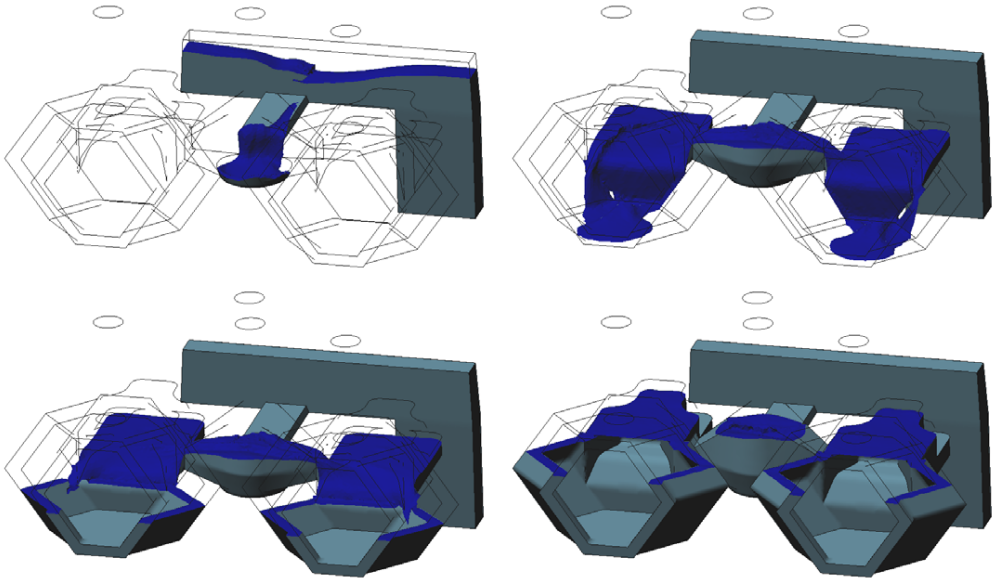


Figure 5.3: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the fine mesh

phase flow monolithic model is used, as we shall show in Section 5.4.

In Figure 5.5 the results on the coarse mesh with an increased Smagorinsky constant ($C^2 = 0.2$) are presented. It can be observed that the results are not significantly affected by the change in the Smagorinsky constant.

	Total	Matrix N. Stokes	Solver N. Stokes
Coarse mesh, $C^2 = 0.2$	$8638s$	69.5%	18.9%
Coarse mesh, $C^2 = 0.05$	$12593s$	69.6%	20.2%
Fine mesh, $C^2 = 0.05$	$90038s$	45.2%	45.6%

Table 5.1: Cpu time

In Table 5.1 the computational times for the previous simulations are presented. It can be observed that the solution of the Navier Stokes equations requires most of the time. For the coarse mesh the assembly of the matrix takes approximately three times more than the solution of the linear system. With the fine mesh both tasks take approximately

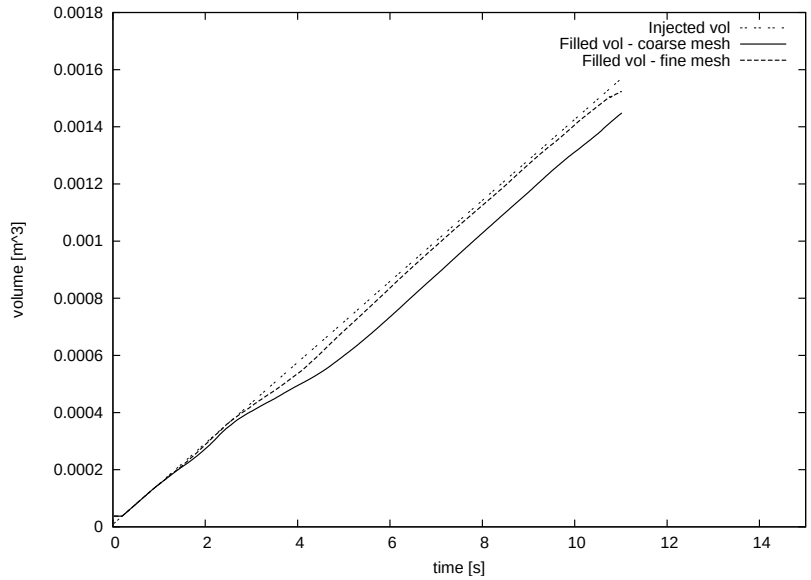


Figure 5.4: Filled volume vs. injected volume for both meshes

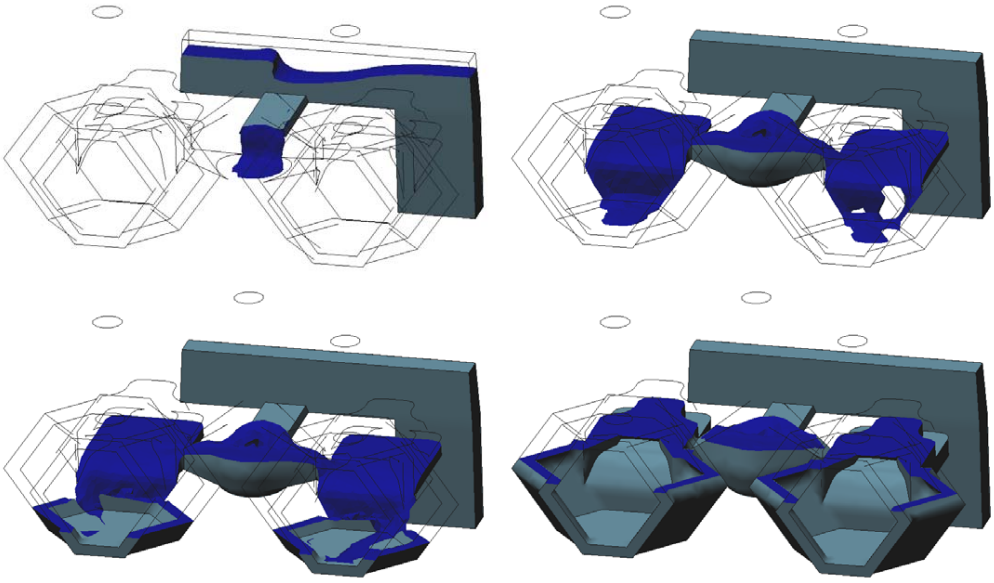


Figure 5.5: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the coarse mesh with a higher Smagorinsky constant

the same time.

Comparison with a commercial code

In order to have some idea on the efficiency of our code we compare the results we have obtained with our model against the results obtained with the commercial code Vulcan [118]. Vulcan uses a fixed mesh finite element pressure correction predictor corrector scheme.

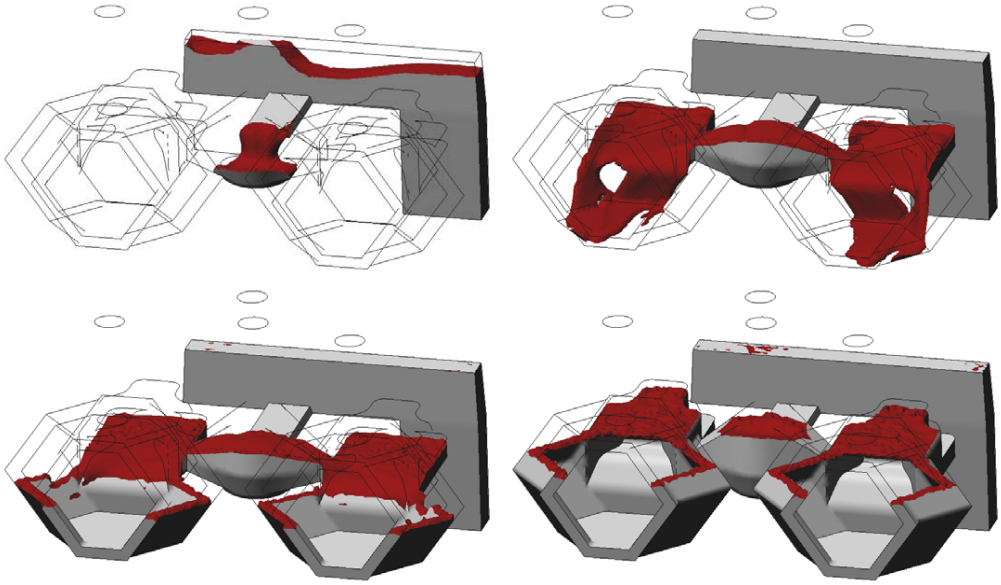


Figure 5.6: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the commercial code with the fine mesh

The results obtained with Vulcan on the fine mesh are shown in Figure 5.6. They are quite similar to the ones we have obtained with our code. The most noticeable difference is the shape of the interface in the last to two figures. With our model a nearly flat surface is obtained. Instead with the commercial codes spurious oscillations can be observed. The enhanced behavior of our code can be attributed to the correct treatment of boundary conditions on the interface.

With the coarse mesh, the results obtained with Vulcan show bigger spurious oscillations (Figure 5.7). Moreover the effect of numerical viscosity becomes quite

noticeable. Instead, with our code, the results obtained on the coarse mesh were much closer to the ones obtained in the fine mesh. As we have already mentioned, the correct behavior of our code even on coarse meshes is a very important feature for mould filing simulations where the complexity of real pieces inhibits the possibility of using too fine meshes.

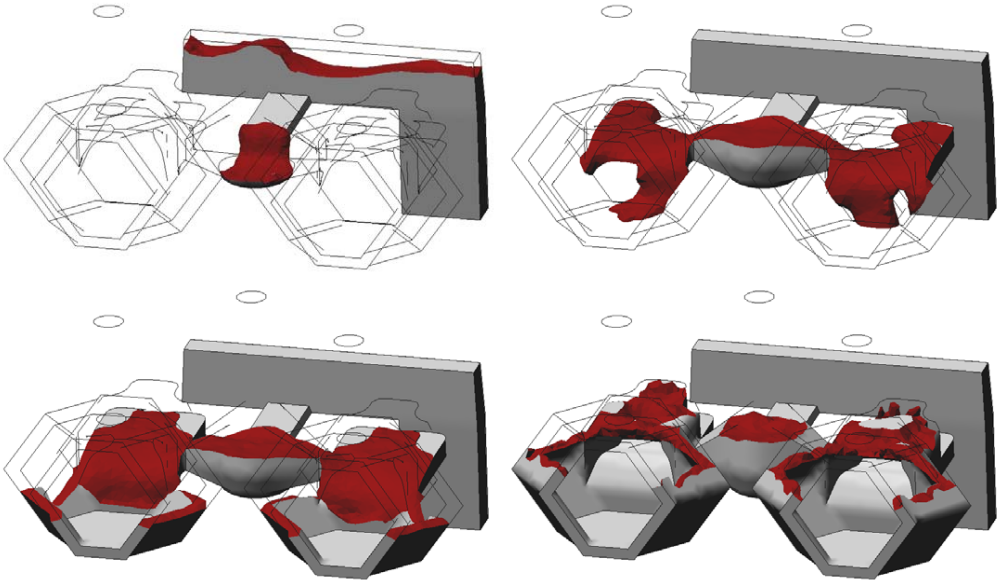


Figure 5.7: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the commercial code with the coarse mesh

The comparison of the computational times obtained with Vulcan against the ones obtained with our code shows that despite our code is only an academic version it can provide competitive results. For the simulation on the fine mesh Vulcan takes nearly twice the time needed by our code. With the coarse mesh it takes 3.2 times more than our code, making the advantage even more notorious. In the simulations with our code we have used a fixed time step size that results in a total of 550 steps. Instead Vulcan uses a variable time step size. A total of 815 and 1225 step have been required on the fine and coarse meshes respectively. The increase in the number of steps with the coarse mesh observed with Vulcan may be related to the increase of the spurious oscillations close to the interface.

5.2.2 Alloy wheel

For the automotive alloy wheel the time steps size we have used is 0.01 seconds and the Smagorinsky constant is $C^2 = 0.05$. In Figure 5.8 the evolution of the interface for different time steps is presented. For the first time step the whole domain is shown and for the remaining steps only the details at the wheel are shown. Once the molten metal reaches the top of the filling tube it slides through the bottom of the spokes until it reaches the their end. Then it turns and fills the lower part of the wheel. Finally it raises through the vertical walls of the wheel. Simultaneously the filling of the spokes is completed. Since we are using a free surface model air is not taken into account so there is no possibility for the formation of air bubbles as happens when the enriched pressure two phase flow model is used (Section 5.4). The possibility of modifying the free surface model so that it can take into account the formation of bubbles will be discussed at the end of the Chapter. Moulds can be classified into two groups depending on whether they allow air to escape though their walls or not. When the air is allowed to escape, such as in sand moulds, the importance of taking into account its effect is less important.

The pressure contour lines are presented in Figure 5.9. A fixed scale with a maximum of 5000 N/m^2 has been used to focus on the pressures inside the wheel. In the filling tube the pressure is nearly hydrostatic but due to the scale we have used it is not shown.

Using a higher Smagorinsky constant ($C^2 = 0.2$) very similar results have been obtained. The higher viscosity imposes an small increase of resistance to the flow. As this piece is filled by an imposed pressure this results in a slightly retarded front. The results have not been shown because the difference is hard to appreciate.

The total CPU time for the simulation has been 38579 seconds. As in the previous example, the resolution of the Navier Stokes equations took most of the time, with 18220 seconds for the matrix assembly and 15470 seconds for the linear solver.

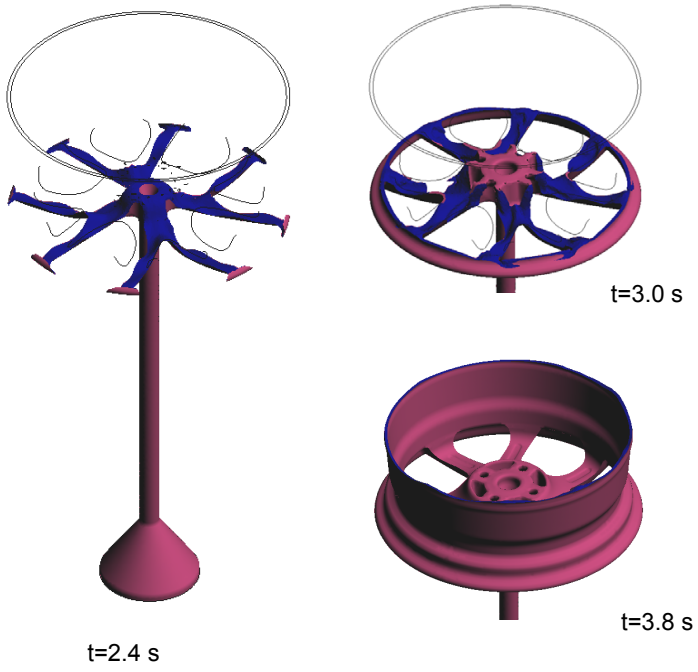


Figure 5.8: Interface evolution at t=2.4, 3.0 and 3.4 *s*

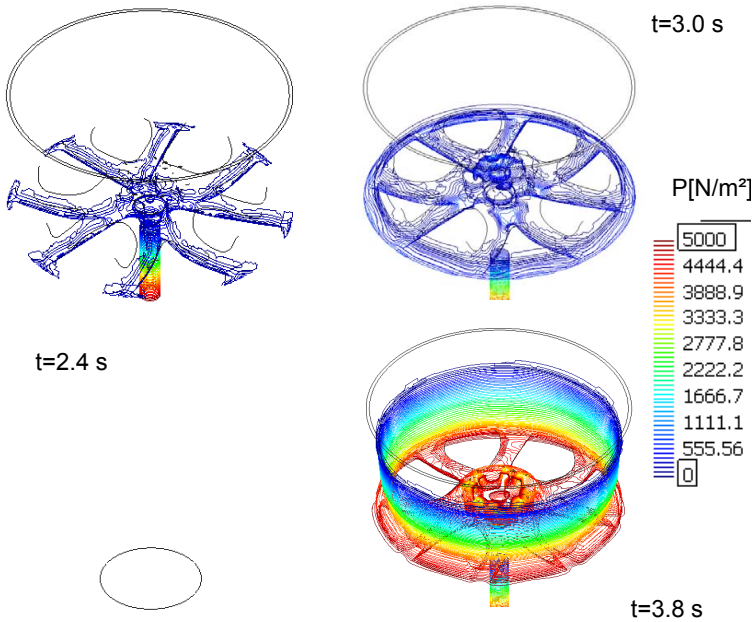


Figure 5.9: Pressures at t=2.4, 3.0 and 3.4 *s*

5.2.3 Shovel

For the shovel a 0.01 seconds time step has been used with two Smagorinsky constants, $C^2 = 0.05$ and $C^2 = 0.2$. Contrary to what happens in the other two pieces, for this case the effect of the Smagorinsky constant is more noticeable, at least during part of the simulation.

Figure 5.10 shows the evolution of the interface for selected time steps using $C^2 = 0.2$. The filling channel used for this piece splits into two branches. One of the branches is closer to the inlet than the other one. As the interface reaches the first branch the molten metal starts falling through it. Approximately one second takes place before the flow starts falling through the second branch. Therefore the side of the shovel closer to the first branch is filled earlier than the part connected to the second branch. When the molten metal exits each of the two branches it slides into two circular parts with a hole in the middle located beneath the end of each branch. Once each of these two parts are full the flow spreads through the base of the shovel and finally raise along the lateral walls to complete the filling process.

For $C^2 = 0.05$ the filling is similar to the one with $C^2 = 0.2$ except when the molten metal goes into the circular part with a hole in the middle. When the lower constant is used one portion of the flow deposits in the bottom of the circular part and the other surrounds the hole in the middle. Instead, when the higher constant is used all of the flow deposits in the bottom of the circular part. A comparison of the filling of the circular part with the two Smagorinsky constants is shown in Figure 5.11. Using a finer mesh it has been observed that part of the flow surrounds the hole in the middle (Figure 5.12). In this sense the results with the lower constant seem better. Unfortunately the flow becomes too complex for the mesh when the smaller constant is used and an important mass loss is introduced as we shall show in Figure 5.13. The mass loss when the smaller constant is used occurs mainly during $t = 3.5\ s$ to $t = 7.0\ s$, the time needed to fill the circular part. With the higher Smagorinsky constant the mass loss is significantly reduced and therefore the results can be preferred despite the lack of precision in the filling of one of

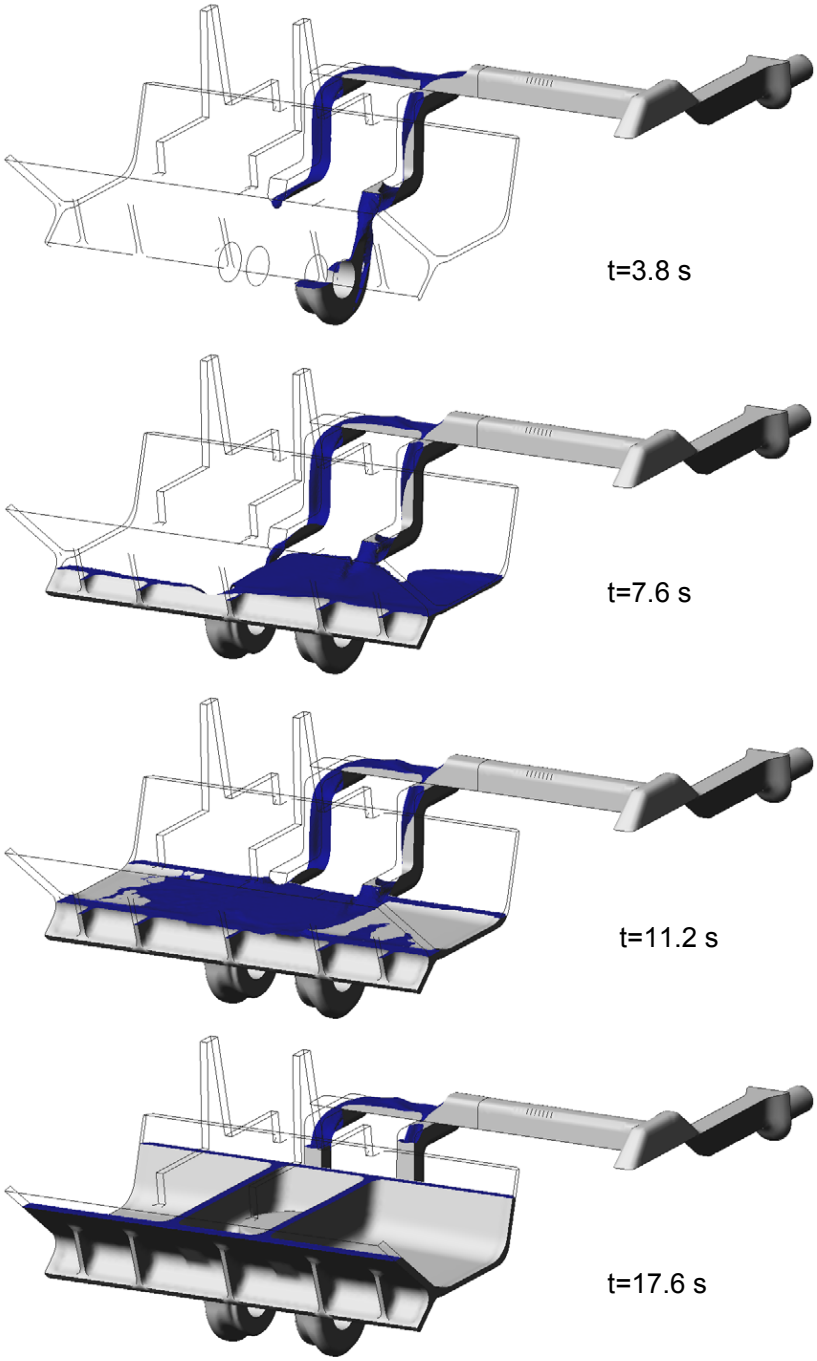


Figure 5.10: Interface position using $C^2 = 0.2$

the circular parts.

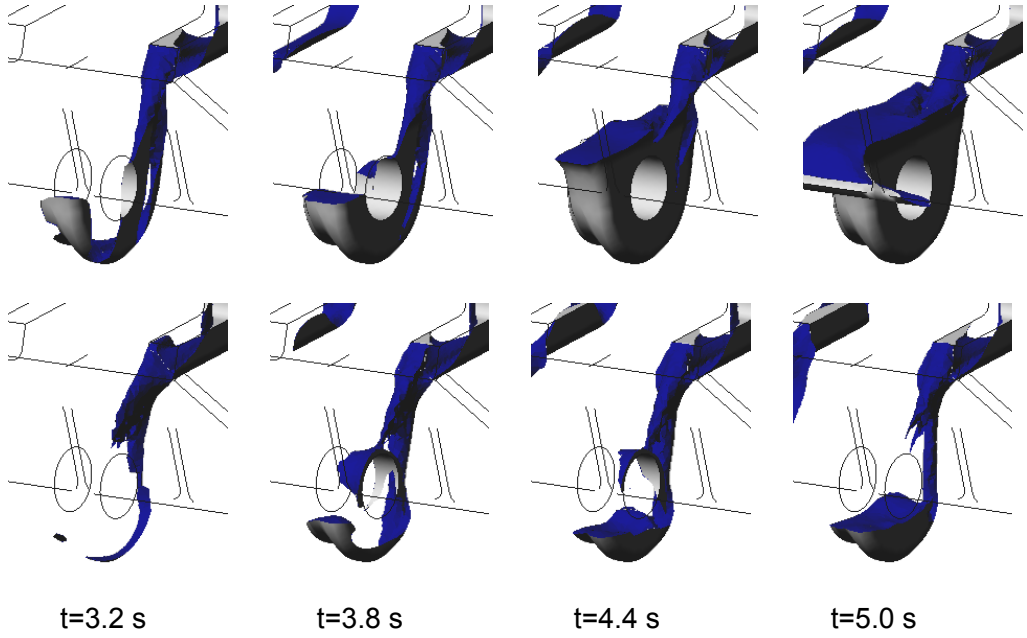


Figure 5.11: Detail of the interface position using $C^2 = 0.2$ (top) and $C^2 = 0.05$ (bottom)

Taking into account that this is a complex piece we have decided to simulate it with a fine mesh formed by 1619428 tetrahedral elements and 319052 nodes. The same time step as in the coarse mesh and a $C^2 = 0.2$ Smagorinsky constant have been used. The results have served as a reference to compare the results obtained with the original mesh. Moreover the simulation on the fine mesh has been used to explore the current limits of our model on a typical PC. The results are shown in Figure 5.12.

In Figure 5.13 we compare temporal evolution of the injected and filled volumes for the previous three cases: coarse mesh with $C^2 = 0.05$ and $C^2 = 0.2$ and fine mesh with $C^2 = 0.2$. As we have already mentioned, the mass loss concentrates mainly during the filling of the two circular parts located at the bottom of the shovel and it is significantly higher when the coarse mesh with the low Smagorinsky constant is used. During the rest of the simulation very little mass loss is observed for the three cases. It is interesting to note that the mass loss is very similar for the two meshes when $C^2 = 0.2$ is used.

In Table 5.2 the CPU times for the shovel simulations are presented. For the fine

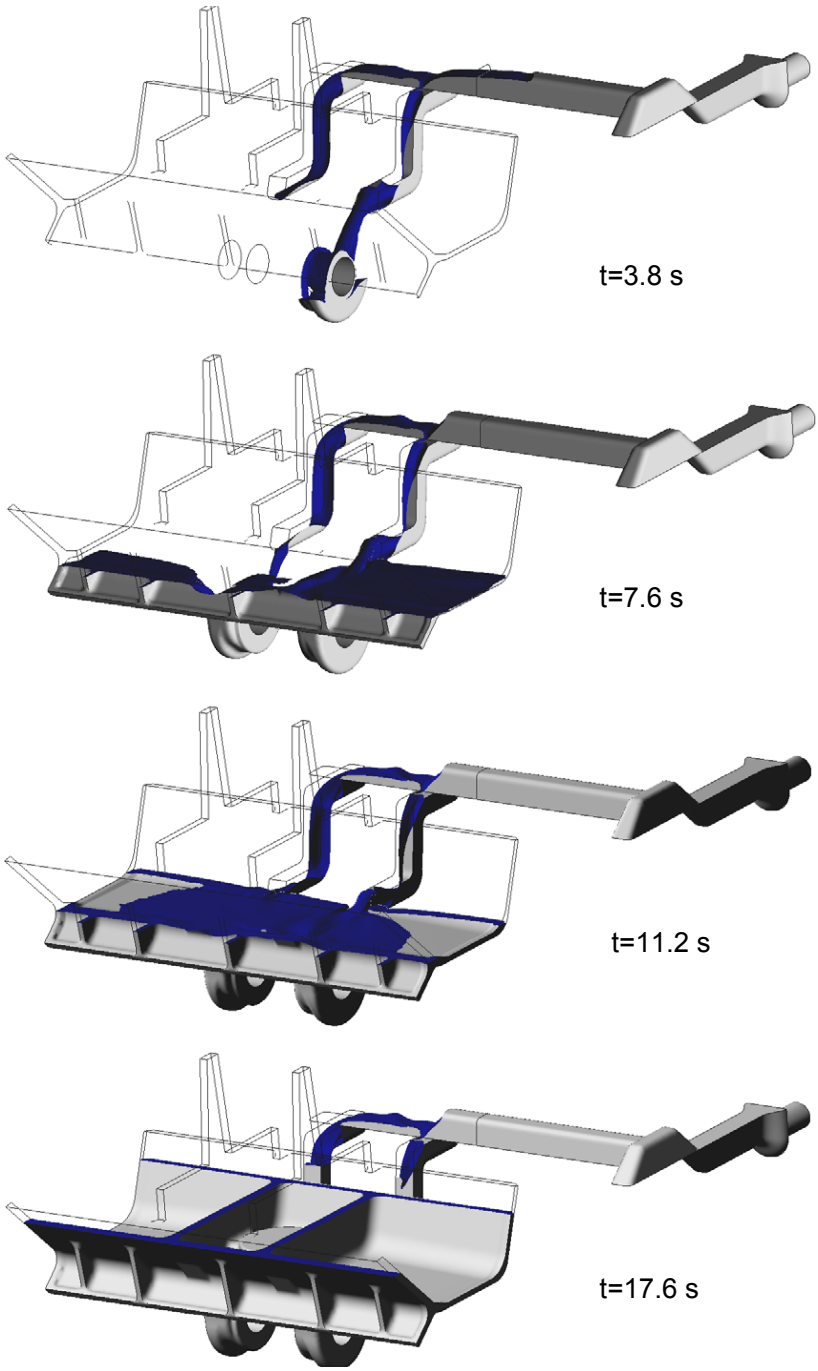


Figure 5.12: Interface position using the fine mesh

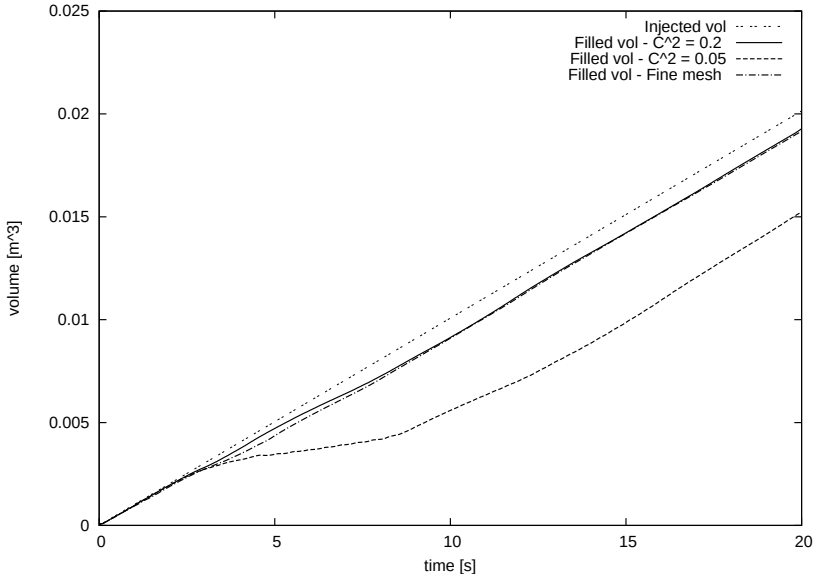


Figure 5.13: Filled volume vs. injected volume

	Total	Matrix N. Stokes	Solver N. Stokes
Coarse mesh, $C^2 = 0.2$	$271681s$	50.8%	34.8%
Coarse mesh, $C^2 = 0.05$	$371473s$	56.8%	30.6%
Fine mesh, $C^2 = 0.2$	$863309s$	40.1%	39.8%

Table 5.2: Cpu time

mesh the cost of matrix assembly is approximately equal to the cost of the linear solver. For the coarse mesh the matrix assembly is more expensive.

Both for the shovel and the alloy wheel it has not been possible to obtain satisfactory results with the commercial code used for the hollow mechanical piece.

5.2.4 Results with the FM-ALE model

As we have anticipated at the beginning of this Section, for the mould filling examples we have not observed any significant difference between the results obtained with the simplified Eulerian model and those obtained with the FM-ALE model. The same behavior has also been observed in the simpler examples presented in Chapter 3. The variations introduced in the evolution of the free surface or the filled volume by the use of the FM-ALE model instead of the simplified Eulerian model are hardly noticeable and therefore the Figures are not be repeated.

In Table 5.3 we present the CPU times obtained with the FM-ALE so that they can be compared against the ones obtained with the simplified Eulerian model. The computational times for the Navier Stokes matrix assembly and solution of the linear system are similar to the ones obtained with the simplified Eulerian model. The total CPU time increases significantly due to the additional steps required by the FM-ALE model. For the moment little effort has been put into optimizing those steps because as both models produce similar results it has been cheaper to use the simplified Eulerian model.

	Total	Matrix N. Stokes	Solver N. Stokes
Hollow Coarse, $C^2 = 0.05$	$25568s$	44.0%	11.8%
Hollow Fine, $C^2 = 0.05$	$186951s$	26.2%	28.9%
Wheel, $C^2 = 0.05$	$80199s$	24.5%	21.3%
Shovel, $C^2 = 0.2$	$447467s$	30.8%	16.1%

Table 5.3: Cpu time

Despite we have not found significant differences between the results obtained with the FM-ALE model and the simplified Eulerian model we belive that in more demanding examples the FM-ALE model may prove advantageous. In [29] we have extended the application of the FM-ALE model to a wider range of problems, including fluid structure interaction. An example of a cylinder moving in a rectangular domain is presented where the advantage of using the FM-ALE model is easily observed. It is mentioned that a large time step is used in order to observe the improvements introduced by the FM-ALE model.

5.3 Free surface velocity correction model

The use of a velocity correction scheme for the free surface model has resulted in the worse results of all four analyzed options. The reason for the poor behavior of this method is still not clear and further research is needed. In Figure 5.14 we show the advancement of the front for a successful run with the free surface velocity correction model. It corresponds to the hollow mechanical piece with the coarse mesh and a $0.005s$ time step. A high Smagorinsky constant ($C^2 = 0.4$) is used in order to make method more robust. In Chapter 3 two alteratives have been proposed; one is to use a FM-ALE method and the other one is to use simplified Eulerian model. When the monolithic scheme is used both of them give very similar results but the simplified Eulerian model is faster. In the velocity correction case the predictor corrector convergence has been easier (but still complicated) with the FM-ALE method and therefore for the results presented in this section it will always be used. The reason might be that the use of the FM-ALE method provides a better velocity extrapolation.

In order to obtain the results shown in Figure 5.14 very low tolerances have been needed both for the predictor corrector iteration and for the velocity and pressure solvers. A predictor corrector scheme with separate loops for the nonlinear and predictor corrector iterations has been used because we have observed that it leads to a more robust scheme. The predictor corrector iteration, that is the most difficult loop to converge for this

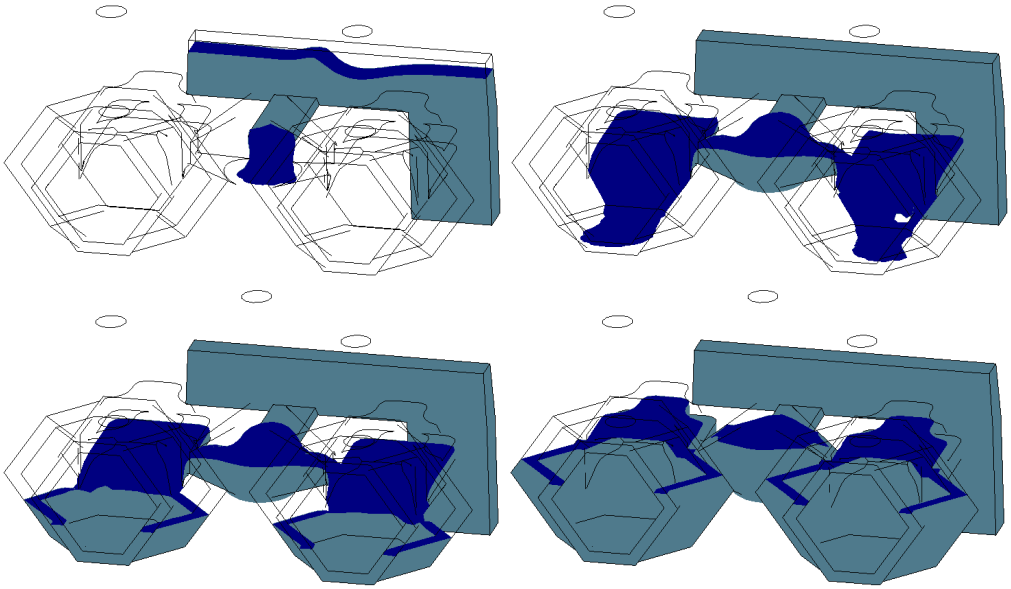


Figure 5.14: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the velocity correction scheme on the coarse mesh

problem, is solved in the inner loop so that error in coupling of the velocity and pressure does not spoil the nonlinear convergence. The stopping criteria for the predictor corrector iteration is set so that the variation of the velocity in the L2 norm is lower than 0.01%. The maximum number of predictor corrector iterations is set to 60. In practice the maximum number of iterations was nearly never reached but during several parts of the simulation more than 10 iterations were used. For the nonlinear iteration a higher tolerance of 1.0% and a maximum of 7 iterations were allowed. The parameters for the nonlinear iteration coincided with those used in a monolithic run used as a reference. For the velocity and pressure solvers a tolerance of 10^{-9} was used.

The combination of a low tolerance for the predictor corrector iteration and for the solvers has made the method very expensive. The total CPU time for this run has been 349398 s. A high percentage of this time, 285343 s, corresponds to the velocity and pressure solvers. For a similar monolithic run the time for the solver is only 14537 s, that is approximately 20 time less. Other alternative runs with higher predictor corrector

or solver tolerances were tested to try to reduce computational times. Some of those runs managed to provide acceptable results for the advancement of the front but showed spurious negative pressures close to the interface. Others would diverge unexpectedly. For the run presented in Figure 5.14 no spurious negative pressures were observed. For an identical run except for the fact that a higher solver tolerance of 10^{-7} was used, spurious pressures appeared at $t = 2.8\ s$ and then at $t = 6.5\ s$ when a 40% of the piece had been filled the results diverged.

In Section 5.5 acceptable results have been obtained with the enriched pressure two phase flow surface velocity correction model without the need to use such low tolerances. Therefore the poor behavior observed with the free surface model seems to be related to the combination of the free surface model, that works very well in the monolithic case, with the velocity correction scheme. The good results obtained with very low tolerances indicate that the problem is not related to an error in the implementation of the free surface velocity correction model.

The spurious pressures that appear close to the interface might indicate that the velocity extrapolation used by the velocity correction scheme close to the interface is poor. The velocities in the region that becomes part of the fluid at each time step can be better extrapolated when the FM-ALE version is used than when the simplified Eulerian is used. The fact that a better behavior is observed when the velocity correction is used with the FM-ALE method can be an evidence that better extrapolations velocities close to the interface help to improve the solution.

A strategy to improve the extrapolation velocities close to the interface could make the free surface velocity correction model more feasible. We propose to improve $\widetilde{\mathsf{U}}_q^{n+1}$ prior to starting each time step. We can call the improved extrapolation $\widetilde{\widetilde{\mathsf{U}}}_q^{n+1}$. It differs from $\widetilde{\mathsf{U}}_q^{n+1}$ only in a small region close to the interface. This region is formed by only 2 or 3 layers of elements. In the nodes from the small region close to the interface in contact with the region where the velocities are not modified the velocities are prescribed to the values $\widetilde{\mathsf{U}}_q^{n+1}$. Then the Navier Stokes equations are solved only in the small region to obtain $\widetilde{\widetilde{\mathsf{U}}}_q^{n+1}$. Since only a small region is being solved, one can expect convergence

of the iterative solver to be very easy. Moreover even if a low tolerance is required for the predictor corrector iteration the computational cost should not increase significantly because only a small region is being solved. A monolithic solver might even be used in this small region.

For the finer mesh and for the other two examples poor results were also obtained with the free surface velocity correction model. In those cases the cost of using very low tolerances makes the simulations too expensive. Therefore we believe that a better strategy should first be obtained with the coarse mesh before stepping to the bigger examples.

5.4 Enriched pressure two phase flow monolithic model

In this section we repeat the previous three examples using the enriched pressure two phase flow monolithic model. The general flow pattern remains very similar to the one obtained with the free surface model but some interesting differences can be observed. Despite some moulds have walls that allow air to escape, in our examples we have supposed that air is only allowed to escape through specified outlets. Therefore regions of entrapped air can be observed when the two phase flow model is used.

As we have observed in the previous examples, mass conservation can be an indicator of the accuracy of the results. With the enriched pressure two phase flow model mass conservation is poorer than with the free surface model. Moreover we have observed that typically mass is lost when ASGS stabilization is used and gained when OSS is used. Instead, a we have already mentioned, with the free surface model no significant dependence on the stabilization method has been observed.

The numerical strategy is the same as for free surface monolithic model.

5.4.1 Hollow mechanical piece

Figure 5.15 shows the evolution of the interface obtained on the coarse mesh with ASGS stabilization, a $C^2 = 0.4$ Smagorinsky constant and a 0.005 seconds time step. Compared to the free surface model case, we have needed a higher Smagorinsky constant and a smaller time step. When the same Smagorinsky constant and time step as in the free surface case is used, the mass loss is too big and the results do not have much interest. Even with the parameters we have used, the mass loss is higher than the one observed with the free surface model with more demanding parameters. However, the evolution of interface is similar to the one obtained with the free surface model but retarded due to the mass loss. In the upper part of the filling channel entrapped air can be observed. This is obviously different to what happens with the free surface model where no air entrapment can occur because air is not simulated.

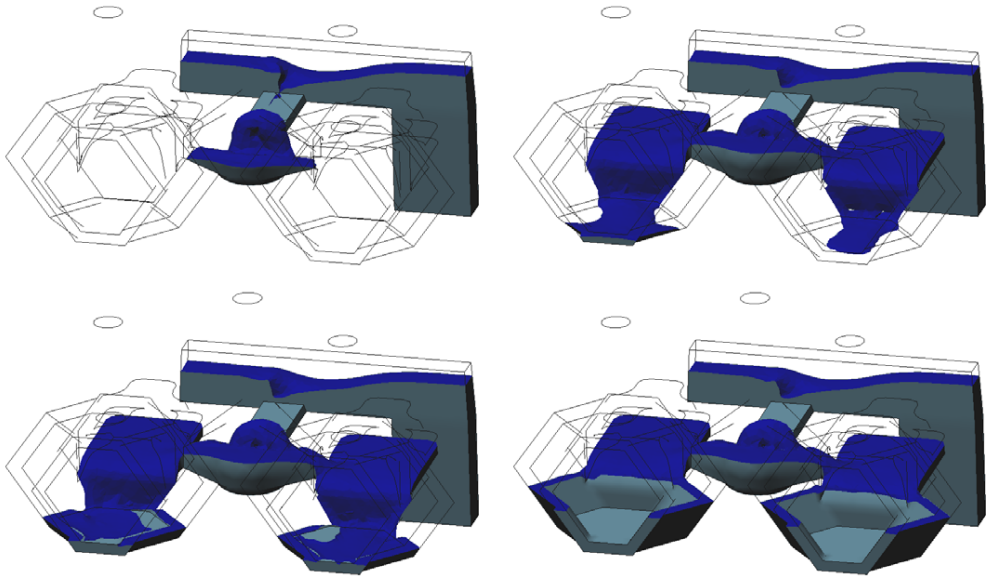


Figure 5.15: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the coarse mesh with ASGS stabilization, $C^2 = 0.4$ and $\delta t = 0.005$ s

With OSS stabilization, both split and non split, two main differences have been observed. Mass conservation is improved; actually there is some mass gain but it is smaller than the mass loss observed with ASGS. On the other hand, less viscosity is introduced

so the nonlinear convergence becomes more difficult and it is hard for some cases to converge. In order to improve the nonlinear convergence anisotropic shock capturing [22] has been introduced. With the coarse mesh we have been able to use the same time step (0.02 s) and Smagorinsky constant ($C^2 = 0.05$) as in the free surface case. Moreover very little dependence on the Smagorinsky constant has been observed. The evolution of the interface and mass conservation obtained with $C^2 = 0.2$ are very close to those obtained with $C^2 = 0.05$. In Figure 5.16 the evolution of the interface obtained on the coarse mesh with split OSS stabilization and $C^2 = 0.05$ is presented. As in the ASGS case the results are similar to the ones obtained with the free surface model but the interface is advanced (instead of retarded) due to the mass gain.

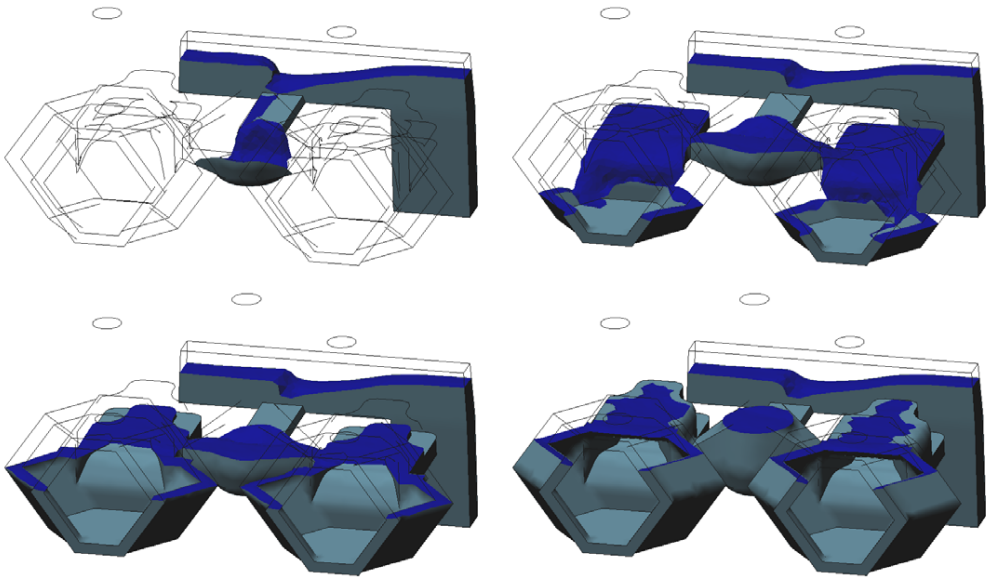


Figure 5.16: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the coarse mesh with OSS stabilization, $C^2 = 0.05$ and $\delta t = 0.02$ s

Figure 5.17 shows the evolution of the filled and injected volumes obtained with ASGS ($C^2 = 0.4$, $\delta t = 0.005$ s) and split OSS ($C^2 = 0.05$, $\delta t = 0.02$ s) stabilization. In order to show how mass conservation degrades with ASGS stabilization the filled volume obtained with $C^2 = 0.1$, $\delta t = 0.01$ s is also included in the comparison.

The fluid mass loss comes from several sources. Despite we are solving the

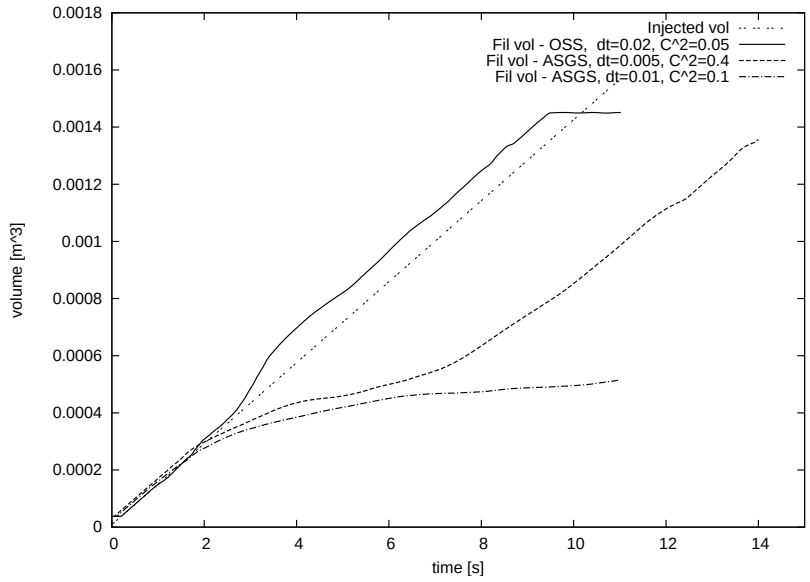


Figure 5.17: Filled volume vs. injected volume for the coarse mesh

incompressible Navier Stokes equations the numerical results are not exactly divergence free. Since our meshes are quite coarse there will be errors in the satisfaction of both the continuity and momentum equations. Pressure stabilization also affects the satisfaction of the incompressibility condition. The errors in the transport of the Level Set function can also cause fluid loss. After solving the level set function it needs to be reinitialized; this may also introduce errors. There might also be some coupling between the previous sources of error.

When the fine mesh is used the mass loss obtained with ASGS stabilization is reduced. Results obtained with $C^2 = 0.1$ and $\delta t = 0.01\ s$ are shown in Figure 5.18. Some mass loss can be observed but the evolution of the interface is acceptable. When the Smagorinsky constant is increased to $C^2 = 0.4$ and the time step is reduced to $0.005\ s$ mass conservation improves as shown in Figure 5.19.

When split OSS stabilization is used with the parameters used for the coarse mesh, $C^2 = 0.05$ and $\delta t = 0.02\ s$ the run diverges. Increasing the Smagorinsky constant to $C^2 = 0.1$ and reducing the time step to $0.01\ s$ good results have been obtained. The evolution of the filled volume is presented in Figure 5.19. The behavior is the same as the

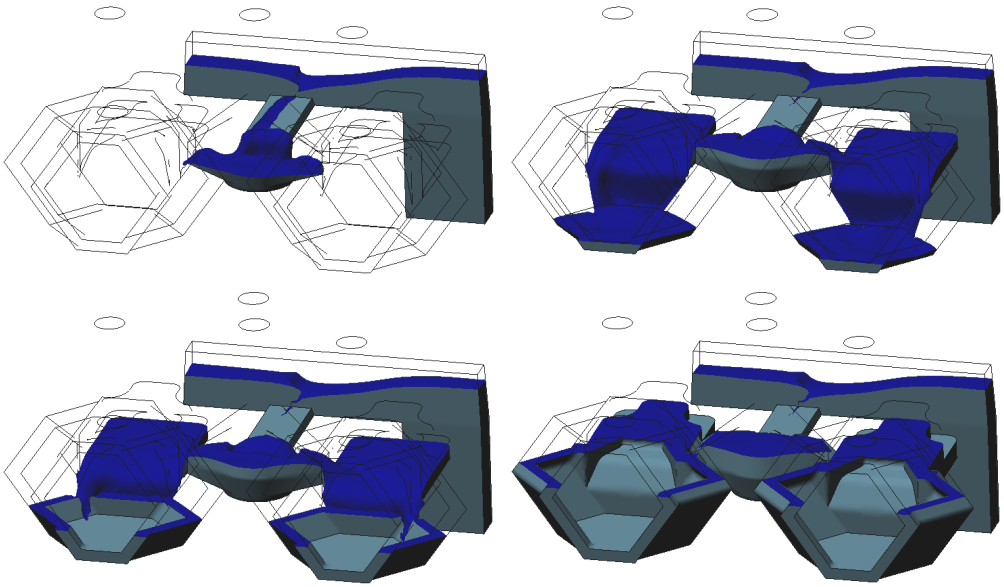


Figure 5.18: Interface position at $t = 1.6$, 3.2, 4.8 and 7.6 s using the fine mesh with ASGS stabilization, $C^2 = 0.1$ and $\delta t = 0.01$ s

one observed with the coarse mesh but the errors are significantly smaller when ASGS stabilization is used.

The flow pattern during the filling process is also important to the foundry person. For example, regions of high velocities can lead to premature mould wear and should be avoided. Figure 5.20 shows the flow field at different time steps obtained with the fine mesh and ASGS stabilization.

We believe that the effectiveness of the method we propose depends strongly on the pressure enrichment we introduced in Chapter 2. In order to prove this we have run the problem with $C^2 = 0.4$ and $\delta t = 0.005$ s on the fine mesh without using the pressure enrichment. The results are much poorer than those shown previously. By the time the filled volume fraction reaches a 14 percent of the mould the mass loss is so important that most of the injected fluid is being lost numerically. The results lose any sense and therefore the runs was stopped. The evolution of the filled and injected volumes is shown in Figure 5.21. It can be observed that the mass loss is similar with ASGS and OSS

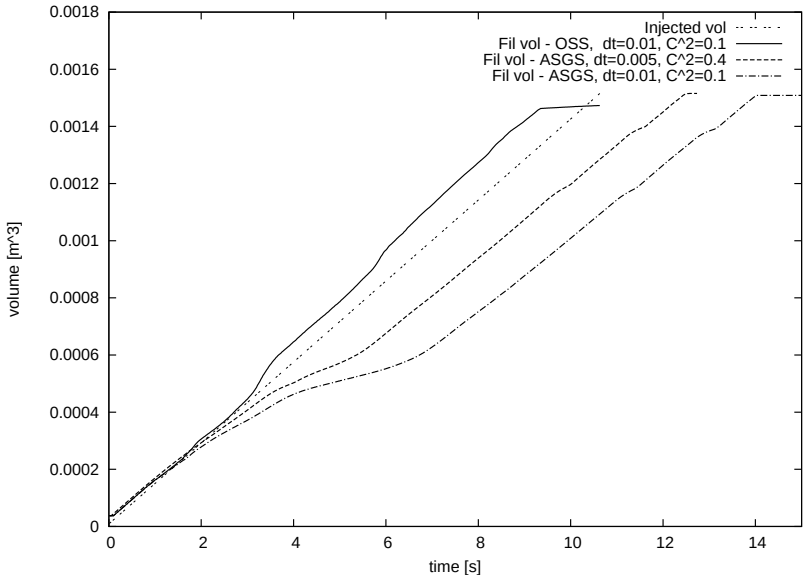


Figure 5.19: Filled volume vs. injected volume for the fine mesh

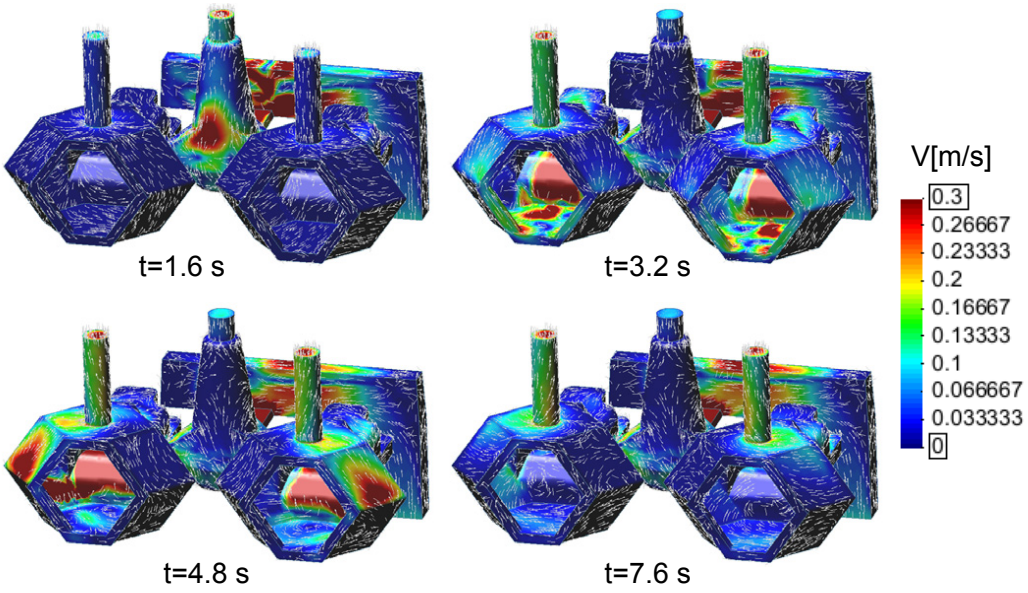


Figure 5.20: Velocity field at $t = 1.6$, 3.2, 4.8 and 7.6 s using the fine mesh with ASGS stabilization, $C^2 = 0.1$ and $\delta t = 0.01$ s

stabilization but in the OSS case spurious velocities appear that lead to oscillations in the filled volume.

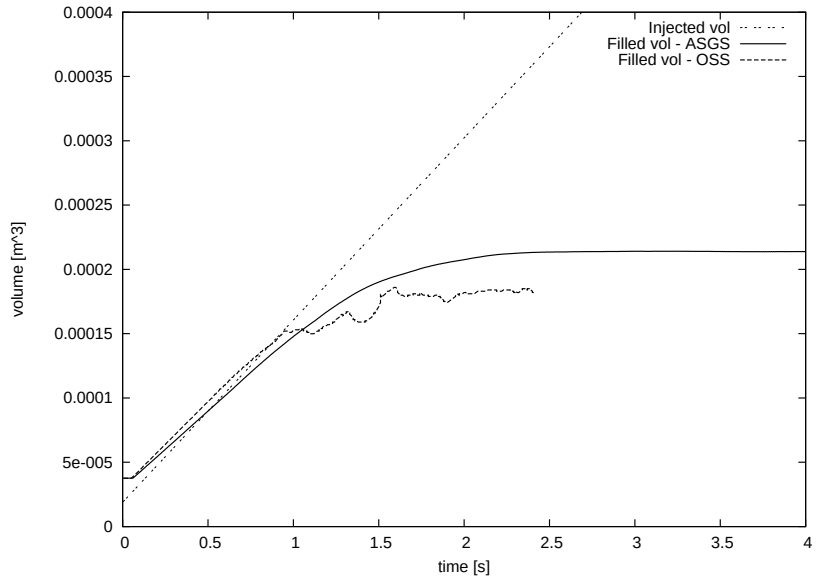


Figure 5.21: Filled volume vs. injected volume without using enrichment

	Total	Mat. N. Stokes	Solv. N. Stokes
Coarse mesh, OSS, $C^2 = 0.05$, $\delta t = 0.02\ s$	$26218s$	74.4%	21.0%
Coarse mesh, ASGS, $C^2 = 0.4$, $\delta t = 0.005\ s$	$82434s$	59.2%	35.9%
Fine mesh, OSS, $C^2 = 0.1$, $\delta t = 0.01\ s$	$457920s$	62.8%	31.6%
Fine mesh, ASGS, $C^2 = 0.1$, $\delta t = 0.01\ s$	$343364s$	51.2%	44.3%
Fine mesh, ASGS, $C^2 = 0.4$, $\delta t = 0.005\ s$	$642212s$	39.3%	44.9%

Table 5.4: Cpu time

In Table 5.4 the computational times for the previous simulations are presented. As in the free surface case, the solution of the Navier Stokes equations requires most of the time. With the coarse mesh and OSS stabilization we have been able to run with the same time step and Smagorinsky constant as in the free surface case. Therefore it is interesting to compare the computational times for this run against the ones obtained

with the free surface model. The total computational time is slightly more than twice the computational time obtained with the free surface model. The free surface model does an average of 2.25 nonlinear iterations per time step and the enriched pressure two phase flow model 2.8, that is approximately 25% more. Therefore only a small part of the advantage of the free surface model comes from the reduced number of nonlinear iterations. Looking at the time needed to compute the Navier Stokes matrix, the free surface model requires less than half of the time required by the two phase flow model. A small part of this advantage can be attributed to the reduced number of nonlinear iterations but most of it comes from the fact that empty elements do not need to be assembled. Since the run starts with a piece that is nearly empty at the beginning of the run the cost of the matrix assembly is very small. Since the run stops when the piece is full, in average between the start and the end of the simulation, approximately only half of the elements need to be assembled when the free surface is used. The time needed by the solver is also approximately twice when the two phase flow model is used. Typically less number of solver iterations are needed per nonlinear iteration when the free surface model is used. Two reasons for this behavior can be given. First, since during part of the simulation the domain that needs to be solved is smaller it seems logical that the matrix is better conditioned. Moreover during most of the simulation the free surface model leads to an non-confined domain. Instead the two phase flow model leads to a nearly confined domain because the flow is only allowed to escape through small air outlets. Typically (not only in interface flows) linear systems arising from the discretization of the Navier Stokes equations on non-confined domains are better conditioned than those obtained on confined or nearly confined domains.

On the coarse mesh, for the ASGS stabilized case, a four times smaller time step has been used to obtain an acceptable mass conservation. The total CPU time is nearly four times bigger than the one obtained with OSS and $\delta t = 0.02\ s$. On the fine mesh, the CPU time for the run with ASGS stabilization and $\delta t = 0.01\ s$ is nearly four times bigger than the time obtained with the free surface model and $\delta t = 0.02\ s$. One possible explanation is that it is twice more costly due to the use of the smaller time step, and twice due to the

use of two phase flow model instead of the free surface model. For the same time step, the results with OSS stabilization take more time than the ones with ASGS stabilization. We believe this can be attributed to the fact that the OSS run needs an average of 2.7 nonlinear iterations per time step and the ASGS run only 1.8. For the ASGS run this time for the solver and for the matrix are similar. Instead for the OSS run, the matrix assembly takes twice more than the solver. We have observed that for this run the solver converges in less iterations when split OSS stabilization is used. Finally the ASGS run with $\delta t = 0.005\ s$ needs nearly twice the time than the one with $\delta t = 0.01\ s$ due to the use of a smaller time step.

We can conclude that for the same conditions two phase flow model takes approximately twice more time than the free surface model. Moreover since the free surface model allows to use bigger time steps to obtain similar results its advantage becomes more notorious.

5.4.2 Wheel

In Figure 5.22 the evolution of the interface obtained with ASGS stabilization, $C^2 = 0.2$ and $\delta t = 0.01\ s$ is presented. For the first time step the whole domain is shown and for the remaining steps only the details at the wheel are shown. It is interesting to see that at some points inside the spokes air is entrapped. This could lead to fabrication defects and should be avoided. At time step $t = 3.8\ s$ an air bubble that is rising to escape as it reaches the upper interface can be seen.

In Figure 5.23 the velocity field for different time steps is presented. Since we are using an inlet pressure that varies linearly with time, while the interface is inside the filling tube the velocities remain quite constant. The increase in the inlet pressure is compensated mainly by an increase in the free surface height. Therefore the position of the free surface raises linearly with time and the velocity in the tube is approximately constant. As the flow enters the wheel and starts sliding down the wheel spokes the increase in the hydrostatic pressure stops but the inlet pressure continues growing linearly. Therefore

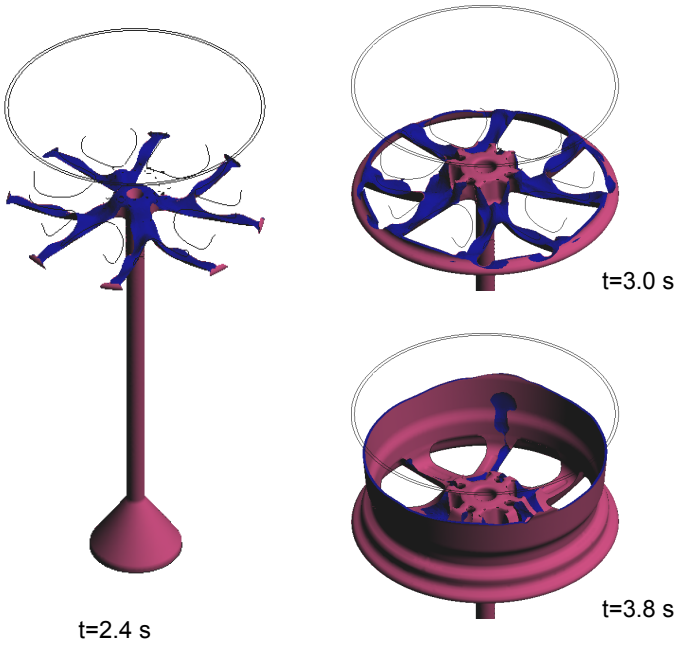


Figure 5.22: Interface evolution at $t = 2.4$, 3.0 and 3.8 s

the flow accelerates until the interface reaches the vertical walls. Finally, the flow rate stabilizes once again until the end of the simulation. At $t = 3.8\ s$ a region of high velocities can be observed at the position where the bubble is located.

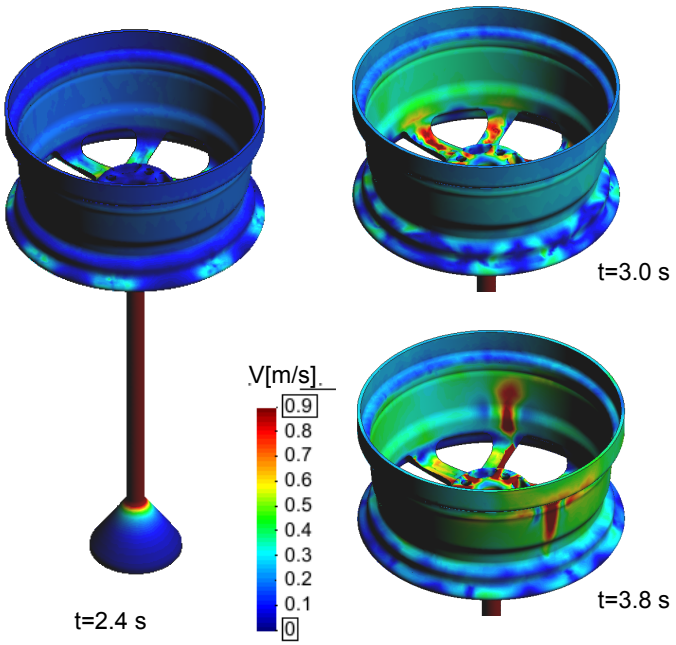


Figure 5.23: Velocity norm at $t = 2.4$, 3.0 and 3.8 s

The total CPU time for this run has been 148418 s. The assembly of the Navier Stokes matrix requires 80760 s and the solver 62541 s. The proportion is similar to the one observed with the free surface model but the times are approximately three times higher despite the same time step is being used.

For this example we have observed that OSS stabilization leads to quite poor results. The nonlinear convergence becomes quite complex and contrary to what happens in the other examples, shock capturing does not help to improve the results. Therefore for this piece we will only present results obtained with ASGS stabilization. It is important to note that the mesh is quite coarse for the flow we are simulating.

As in the previous example, the simulation was also run without using the pressure

enrichment. The results without pressure enrichment are much poorer than those obtained with pressure enrichment. In Figure 5.24 the evolution of the interface for the case without enrichment is shown. Up to $t = 2.4\ s$ the results are similar to those obtained with the enriched model. As the flow starts sliding down the wheel spokes the numerical mass loss becomes much more important than in the case with enrichment. For time $t = 3.0\ s$ the mass loss is easily noticeable and at $t = 3.8\ s$ it is very important.

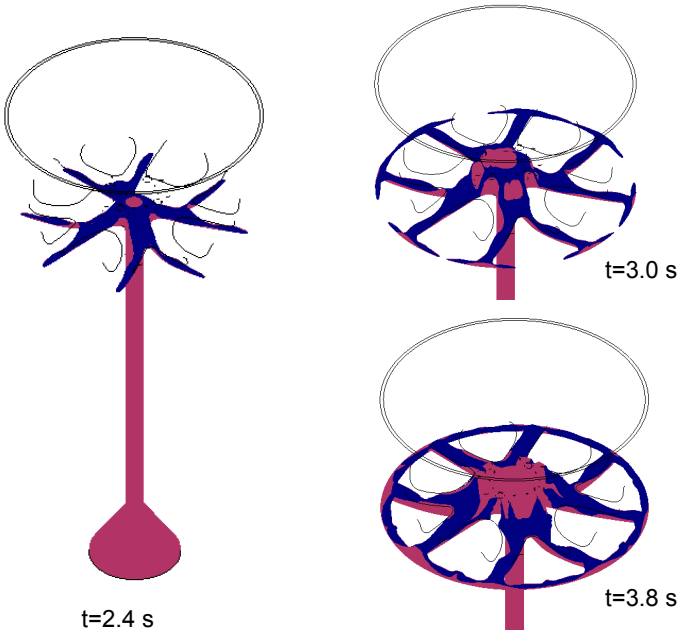


Figure 5.24: Interface evolution for the case without enrichment

5.4.3 Shovel

Figure 5.25 shows the results obtained with ASGS stabilization, $C^2 = 0.4$ and $\delta t = 0.005\ s$. We have used those parameters because when the time step is increased and the Smagorinsky constant decreased the mass loss increases significantly. As in previous pieces, the evolution of the interface is similar to the one obtained with the free surface model but retarded due to the mass loss. When split OSS stabilization with $C^2 = 0.2$ and $\delta t = 0.01\ s$ is used the behavior of the flow is very similar to the one obtained with

ASGS but as mass is gained instead of lost the front is advanced instead of retarded. The same behavior has already been observed in the hollow mechanical piece and therefore the evolution of the interface for the OSS run is not presented for the shovel.

In Figure 5.26 the comparison of the filled and injected volumes is presented. The run with split OSS stabilization, $C^2 = 0.2$ and $\delta t = 0.01\ s$ gains mass and the run with ASGS stabilization, $C^2 = 0.4$ and $\delta t = 0.005\ s$ loses mass. We have also introduced the results obtained with ASGS stabilization, $C^2 = 0.2$ and $\delta t = 0.01\ s$ to show that it introduces an important mass loss.

In the case without enrichment the mass loss is so important that after some time most of the injected fluid is being lost numerically and therefore the runs were stopped. Figure 5.27 presents the comparison of the filled and injected volumes obtained with $C^2 = 0.4$ and $\delta t = 0.005\ s$. Both ASGS and Split OSS stabilized runs show a similar behavior but with OSS stabilization the filled volume stalls at 22% and with ASGS at 17%. The advantage of using pressure enrichment can easily be observed by comparing with the results presented in Figure 5.26.

In Table 5.5 the computational times for the shovel simulations are presented. The run with split OSS stabilization uses the same time step and Smagorinsky constant as one of the runs presented in the free surface monolithic section. The comparison of the CPU time shows that the enriched pressure two phase flow model takes more than twice the time than the free surface model. The time required by the solver for the Navier Stokes equations is three times bigger when the enriched pressure two phase flow model is used. For the ASGS run, the use of a smaller time step makes the CPU time to increase to nearly twice the time needed with OSS stabilization.

	Total	Matrix N. Stokes	Solver N. Stokes
OSS, $C^2 = 0.2$, $\delta t = 0.01\ s$	$590362s$	38.4%	56.3%
ASGS, $C^2 = 0.4$, $\delta t = 0.005\ s$	$1106010s$	37.1%	57.3%

Table 5.5: Cpu time

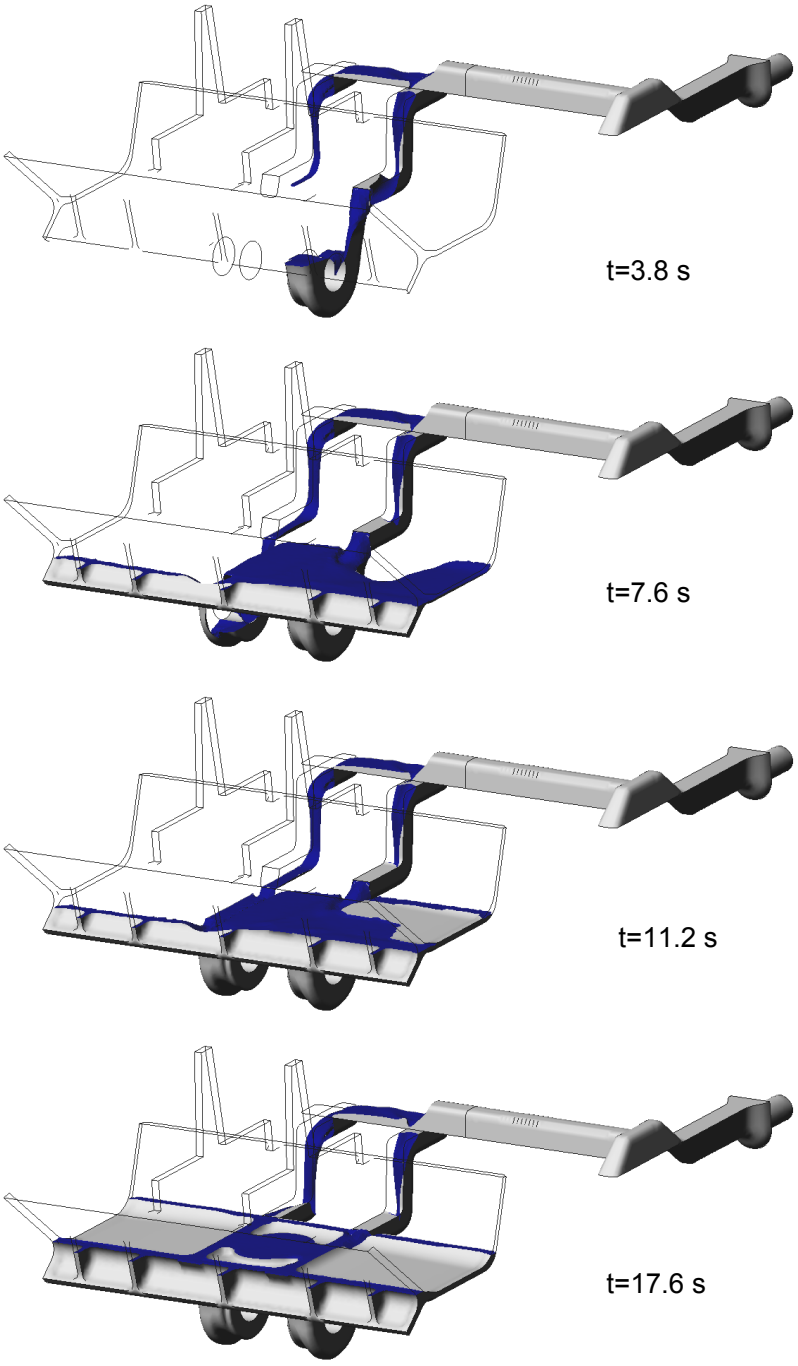


Figure 5.25: Interface position using ASGS stabilization, $C^2 = 0.4$ and $\delta t = 0.005\ s$.

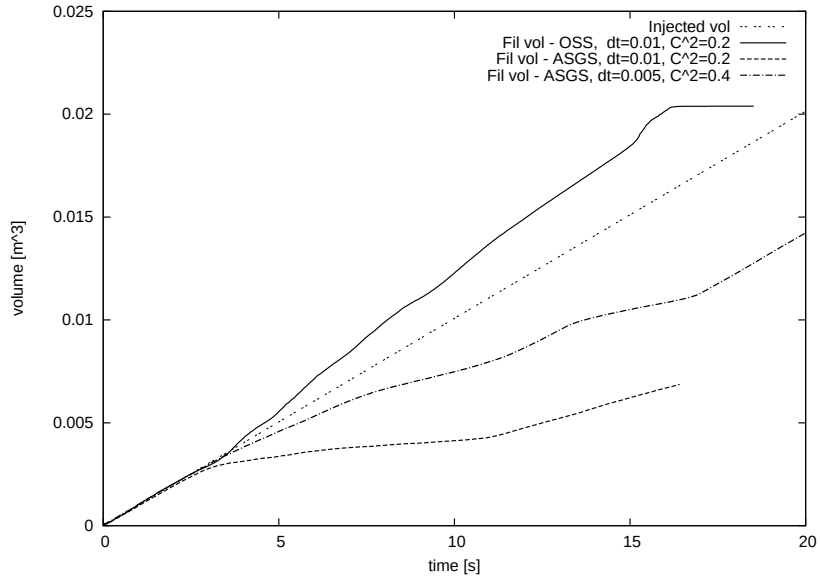


Figure 5.26: Filled volume vs. injected volume

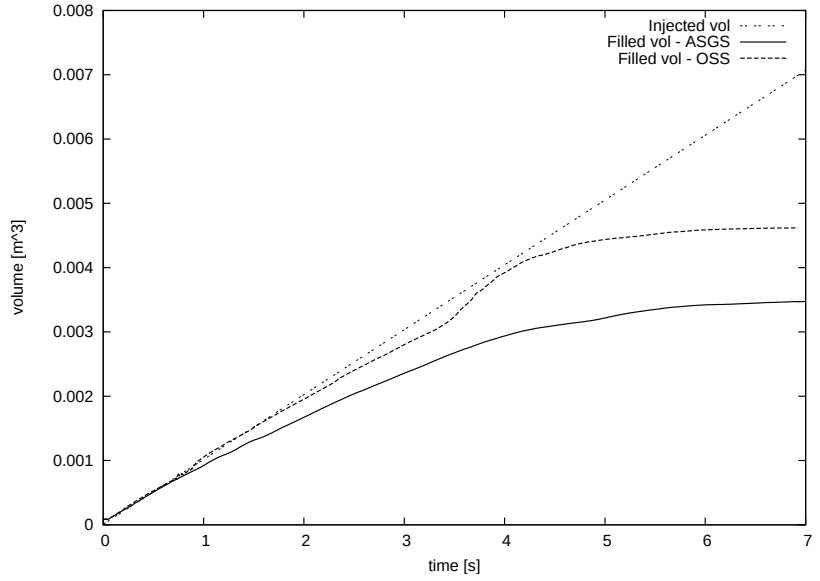


Figure 5.27: Filled volume vs. injected volume for the case without enrichment

5.5 Enriched pressure two phase flow velocity correction model

In this section some of the cases presented in the previous section are repeated with a predictor corrector velocity correction model. The objective is to show that very similar results to the ones presented with the monolithic model can be obtained.

The numerical strategy we have used consists of a external predictor corrector loop with a maximum of 7 iterations and a stopping criteria set so that the variation of the velocity in the L2 norm is lower than 1.0%. The non linearity is dealt in a internal loop with a maximum of 3 iterations and a stopping criteria of 0.8% variation of the velocity in the L2 norm. For the solution of the velocity equations a GMRES solver with a simple LU-SGS [75] preconditioner is used. The stopping criteria used is that the residual falls below 10^{-6} times the right hand side and a maximum of 300 iterations are allowed. For the pressure a discrete Laplacian is used. It is solved with a Conjugate Gradient solver when Split OSS stabilization is used and with a GMRES solver when ASGS is used (see Subsection 4.3.6). No preconditioner is used for the pressure. The same stopping criteria as for the velocity is used. Complex preconditioners have been avoided so that the method can be parallelized in the future.

5.5.1 Hollow mechanical piece

The evolution of the interface obtained with ASGS stabilization, a $C^2 = 0.4$ Smagorinsky constant and a 0.005 seconds time step on the coarse mesh is indistinguishable from the one obtained with the monolithic model and the same parameters (Figure 5.15) and is therefore not repeated. The mass loss also remains unaltered from the one shown for the monolithic model in Figure 5.17.

Instead with Split OSS stabilization the results obtained with the velocity correction method were poorer than the ones obtained with the monolithic model. In the monolithic case OSS stabilization made possible the use of a big time step (0.02 s) and small

Smagorinsky constant ($C^2 = 0.05$). With those parameters the velocity correction run diverged. A 0.002 s time step and $C^2 = 0.4$ Smagorinsky constant were needed to obtain acceptable results.

For the fine mesh the run with ASGS stabilization, a $C^2 = 0.4$ Smagorinsky constant and a 0.005 seconds time step was also repeated with the velocity correction scheme. The results are again very close to the ones obtained with the monolithic model. As in the coarse mesh, the convergence with Split OSS stabilization is difficult and therefore no results are presented.

	Total	Mat. N. Stokes	Solv. N. Stokes
Coarse mesh, ASGS, $C^2 = 0.4$, $\delta t = 0.005$ s	148148s	76.1%	21.3%
Coarse mesh, OSS, $C^2 = 0.4$, $\delta t = 0.002$ s	274638s	54.7%	42.8%
Fine mesh, ASGS, $C^2 = 0.4$, $\delta t = 0.005$ s	910100s	69.8%	27.3%

Table 5.6: Cpu time for the velocity correction runs

In Table 5.6 the computational time for the velocity correction runs is presented. Both ASGS runs (coarse and fine) takes approximately 50% more CPU time than the corresponding monolithic runs. The time for the solver is approximately the same but the matrix assembly takes more than twice the time than the monolithic run. With OSS stabilization the use of a very small time step leads to high CPU times.

5.5.2 Wheel

When the run presented in the monolithic case (ASGS stabilization, $C^2 = 0.2$ and $\delta t = 0.01$ s) was repeated with the velocity correction scheme it diverged. A smaller time step ($\delta t = 0.005$ s) and a bigger Smagorinsky constant ($C^2 = 0.4$) were needed to obtain acceptable results. The evolution of the interface is shown in Figure 5.28. Although the time step and a Smagorinsky constant are not the same as in the monolithic run the results are very similar.

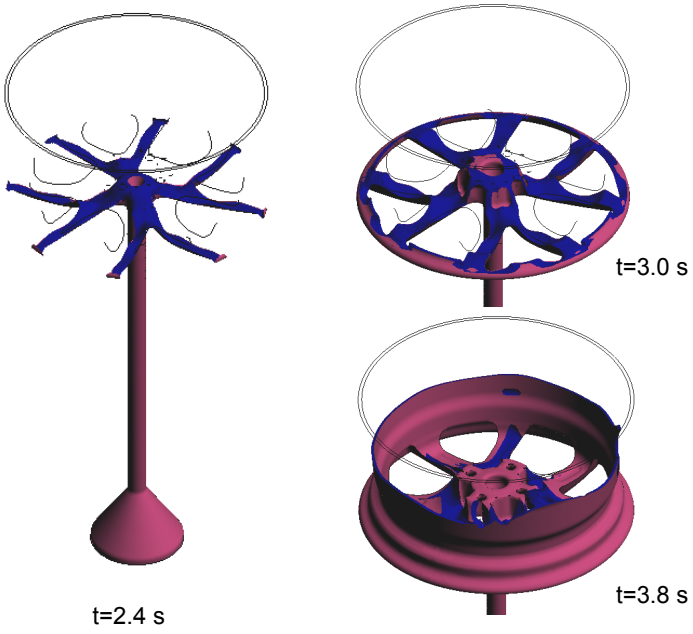


Figure 5.28: Interface evolution at t=2.4, 3.0 and 3.8 s

The total CPU time for this run has been 611339 seconds. The assembly of the Navier Stokes matrix requires 412355 s and the solver 191228 s. The CPU time is four times bigger than in the monolithic case, in part due to the use of a smaller time step.

5.5.3 Shovel

The run with ASGS stabilization, $C^2 = 0.4$ and $\delta t = 0.005$ s presented in the monolithic case was repeated with the velocity correction scheme. The evolution of the interface is very close to the one obtained with the monolithic model and is therefore not repeated. The total CPU time for this run has been 1627990 seconds. The assembly of the Navier Stokes matrix requires 1053970 s and the solver 532796 s. An in the hollow mechanical piece example the total time increases approximately 50% due to the increase in the cost of the matrix assembly. The solver cost decreases approximately 20%.

5.6 Conclusions

In this Chapter we have explored the suitability of methods developed in previous chapters for the simulation of mould filling problems borrowed directly from the foundry. Low Froude number examples have been chosen because we have found that they are typically more difficult to solve than high Froude number examples. They require an accurate treatment in the region close to the interface.

In this thesis, two methods have been proposed to improve the simulation of Low Froude number interface flows: an enriched pressure two phase flows model (Chapter 2) and free surface model (Chapter 3). The results presented in this Chapter show that both of them provide significant improvements over the usual two phase flows model. In the case of the enriched pressure two phase flow model the improvement comes from the enhanced approximation of the discontinuous pressure gradient in the elements cut by the interface. In the free surface model the problem with the discontinuous pressure gradient disappears because only one fluid is simulated. The advantage of the free surface method we propose relies on the accurate imposition of boundary conditions at the interface thanks to the

use of enhanced integration.

The comparison between both models shows that the free surface model provides significantly better results than the enriched pressure two phase flows model. For runs with the same time step and Smagorinsky constant the free surface model provides two or three times lower CPU costs. The main advantage comes from the fact that only the domain filled by fluid is being solved. A typical mould filling simulation that starts with an empty piece and ends when the piece is full. Therefore, in average, the cost of matrix assembly is half of the one required by the enriched two phase model that must solve both the fluid and the air. The cost of the solver is also reduced thanks to the solution of a smaller domain. Moreover it has been observed that with the free surface model bigger time steps and smaller Smagorinsky constants can be used. The combination of the reduced computational time observed for identical runs with the possibility of using bigger time steps leads to a CPU time reduction of nearly an order of magnitude when the free surface model is used.

Additionally we have observed a much better mass conservation when the free surface model is used. The results obtained with the free surface model show little dependence on which stabilization technique is used. Instead when the enriched pressure two phase flows model, not only mass conservation is poorer, but it also depends on which stabilization method is used. With ASGS mass is lost and with OSS it is gained. The reason for this difference should be explored further. The strong dependence of the enriched pressure two phase flows model on the stabilization technique used is a negative aspect of the method.

Both the enriched pressure two phase flows model and free surface model have also been implemented with a velocity correction pressure segregation scheme with the objective of obtaining a more efficient scheme. The use of the velocity correction scheme with the enriched pressure two phase flows model has provided results that are very similar to the ones obtained with the monolithic model. The computational times are higher when the velocity correction method is used but further work in order to reduce the time of the matrix assembly should lead to similar efficiency for both methods. However since we have observed that the enriched pressure two phase flows model is not competitive with the

free surface model for the moment this does not seem to be a priority. The improvement of the efficiency of the velocity correction method used with the free surface model would be more important. Unfortunately this combination is not working satisfactorily for the moment.

Some ideas to improve the behavior of the velocity correction scheme with the free surface model have been proposed. We belive that the correct combination of pressure segregation methods with the free surface model seems very promising in the medium or long term. In the short term, the use of pressure segregation methods does not seem crucial unless further examples where the relative cost of the solver increases are found. We believe the use of a pressure segregation method has two key benefits. The first one is that it leads to better conditioned systems at the cost of uncoupling the velocity and the pressure. This pays off when the cost of solving the linear system becomes dominant. For the examples we have run we have observed that cost of solution of the linear is equal of smaller than the cost of the matrix assembly even in the biggest problems we have run. For bigger problems, the relative weight of solver cost is expected to increase since the cost of the matrix assembly grows linearly with the size of the problem but the cost of the solver grows at a rate greater than one. The solution of bigger problems would need the program to be parallelized because for the moment we have reached the size of problems that can be solved with a serial code. Therefore, as long as the cost of the solver does not become dominant, the interest in the velocity correction scheme is related to the parallelization of the code. The second benefit of using pressure segregation methods is also associated with the parallel implementation. Complex preconditioners are hard to parallelize and it is therefore advantageous to have a method that does not need preconditioning.

The comparison with a commercial code shows that both methods we propose provide better results in the hollow mechanical piece. For the other examples while the methods we propose have provided satisfactory results the commercial code has not. Therefore we can conclude that our methods are more robust than the commercial code. Moreover we have shown that using the free surface model we obtain smaller CPU times than the

commercial code.

When the enriched pressure two phase flows model is used both the fluid and the air are simulated as an incompressible flow. Therefore air bubbles can be formed. Specially in moulds that do not allow air to escape through their walls, the formation of entrapped air regions can lead to defects and should be avoided. In this sense the two phase flows model may the considered better than the free surface model that does not take air into account. In order to overcome this problem the free surface model can be modified to take air into account as is done for example in [77] or [16]. The Navier Stokes equations are solved only in the fluid and in the air bubbles it is assumed that the timescales associated with the speed of sound in the bubble are much faster than the timescales of the surrounding fluid. The pressure in the bubble is therefore spatially constant [77]. In [16] a model that treats air as an ideal gas and can deal with the spliting and merging of bubbles has been developed. Numerical examples are presented where the increase in the CPU time introduced by taking into account the air is less than ten percent. We believe that the possibility of taking into account air in our free surface model could be an interesting enhancement. Moreover, we belive that treating air as an ideal gas might be a better approximation than modelling it as an incompressible flow. In its interaction with a much denser fluid air can undergo significant pressure changes that should result in a volume change something the incompressible flow model can not take into account.

For the problems we are dealing with, the use of relatively coarse meshes is currently unavoidable. The physics is therefore not fully solved and convergence problems may occur. For such problems, the use of artificially high Smagorinsky constants is accepted to make the method more robust.

Chapter 6

Conclusions

Since each of the Chapters in this thesis, except the first one, have their own section devoted to conclusions, in this Chapter we shall simply try to review our most important contributions and discuss future lines of research.

6.1 Achievements

The objective of this thesis is the improvement of two phase flows finite element modeling on fixed meshes and its application to mould filling problems. We believe that our first achievement was the identification of Low Froude number flows on fixed meshes as an area that deserved further research. The problem was brought to us by Professor Buscaglia [82]. He pointed out the impossibility of simulating two different density fluids at rest under gravity forces when the mesh is not aligned with the interface. We then found that the problem extended to all low Froude number flows and that in mould filling simulations such flows were typically the most demanding cases.

The correct representation of the pressure gradient in the elements cut by the front is needed for low Froude number flows. An enriched pressure two phase model is presented in Chapter 2. The impossibility of fixed mesh methods of correctly representing the discontinuous pressure gradient in elements cut by the interface is identified as the key problem for the correct simulation of low Froude number flows. The solution we propose is

to enrich the pressure shape functions in the elements cut the interface. The enrichment is local to each element and can therefore be condensed prior to assembly making the implementation quite simple on any 2D or 3D finite element code. The advantage compared to XFEM methods, that also enrich the shape function in the elements cut by the front, is that no additional degrees of freedom need to be added to the system matrix. Once XFEM is well developed for 3D problems it can be an interesting alternative to our method. In order to take advantage of the enrichment enhanced integration rules that subdivide cut elements according to the position of the interface have been used. The enriched pressure two phase model has been introduced in [37] and further examples have been presented in [36].

As an alternative to the previous model a free surface model on fixed meshes has been developed. By free surface we understand that only one fluid is simulated and the influence of the second fluid on the first one is neglected. There are a wide number of flows where this hypothesis is valid. The simulation of such flows is simpler than two phase flows and therefore we believe it is advantageous to use this model when possible. As only one fluid is simulated the problem with the discontinuous pressure gradient disappears. Actually the possibility of discontinuous velocity gradient also disappears and surface tension could easily be introduced at the free surface. The key ingredient of our free surface method is the use of enhanced integration that allows us to impose Neumann boundary conditions at the interface accurately. The free surface model has been introduced in [38] and further examples have been presented in [39].

Simulating a free surface flow on a fixed mesh introduces the particularity that despite the mesh is fixed the domain that is being simulated is moving. We have extended the FM-ALE approach proposed in [59] to correctly take into account this effect. Moreover in [29] we have generalized the FM-ALE concept to other fields such as fluid structure interaction to correctly take in account the movement of the domain when fixed meshes are used.

We have explored pressure segregation methods with the objective of improving our computational efficiency and facilitating the possibility of a parallel implementation in the

future. Both pressure correction and velocity correction methods have been implemented in our code. The numerical comparison on one phase flows shows that the velocity correction scheme provides some advantages over the pressure correction scheme. The most notorious advantage is the possibility of obtaining a numerically stable third order fractional step scheme. In [6], where the velocity correction method that we use has been introduced, obtaining a third order stable scheme had not been possible. The key difference is that we have used both continuous and discrete Laplacian approximations whereas in [6] only a continuous Laplacian had been used. For the velocity correction fractional step scheme we observe that only the discrete Laplacian allows to obtain a pure velocity correction method in the sense that it is completely independent of the pressure extrapolation. When a continuous Laplacian is used, a second order pressure extrapolation is needed to obtain a third order scheme. This has been identified as the source of the instability and it disappears when a discrete Laplacian is used.

The fact that the velocity correction scheme works better on one phase flows and the observation that the velocity can be better extrapolated than the pressure for interface flows has motivated us to use velocity correction schemes for such flows. For the moment the results are not as satisfactory as we would have desired. For the interface problems we are dealing with the monolithic solver has turned out to be more efficient. Actually, initially we had expected the monolithic system to be harder to solve than what we have found. With the enriched pressure two phase model the velocity correction has provided satisfactory results but somehow slower than the monolithic version. The combination with the free surface model still requires further work.

Both the enriched pressure two phase model and the free surface model have been successfully applied to complex mould filling problems. The advantages they introduce in low Froude number flows have also been verified in mould filling problems by comparing with a commercial code. The free surface model has proven to be a more efficient option. Not only does it provide lower ($\approx 50\%$) CPU time than the enriched pressure two phase model, but it also allows to use bigger time steps (and lower Smagorinsky constants) leading to efficiency advantages of nearly an order of magnitude. Despite our code is only

an academic version we have shown it can provide both better results and computational times than the commercial code.

For the solution of real mould filling problems, where the use of wall laws is mandatory, we found problems with non Dirichlet curved boundaries under gravity forces. We developed a solution for this problem that uses 'do nothing' boundary conditions. Actually we then found out that a very similar strategy had recently been proposed in [9]. The advantage of our method is that it introduces no modification on planar boundaries.

Both ASGS and OSS stabilization have been used for two phase flow problems. In the OSS case we have found that the straight forward application of what is done in one phase flows leads to very poor results. An enhancement that takes into account the density variation in the projection of the residual has been introduced.

6.2 Open lines of research

As we have already anticipated in the achievements Section, other methods could also be used to improve the representation of the pressure in cut elements. The XFEM method seems to be the most popular option, but alternatives such as the one proposed in [53] may also be extended to two phase flows. Moreover we believe the ideas developed in [27] could be combined with the formulation proposed in [53] to obtain an improved method.

Despite we have concentrated in improving the representation of the discontinuous pressure gradient because we have found that its misrepresentation has the greatest effect on the modeling of low Froude number flows, other discontinuities exist at the interface. Surface tension introduces a discontinuity in the pressure that could be represented using a discontinuous pressure enrichment. Even in the case without surface tension the pressure can be discontinuous due to the discontinuity in the viscous terms. We have preliminarily explored the use of a discontinuous velocity gradient enrichment but the advantages it introduces are much smaller than those introduced by the pressure enrichment.

The combination of the velocity correction scheme with the enriched pressure two phase model has provided satisfactory results but the efficiency must still be improved so

that it can be competitive against the monolithic model. The combination with the free surface model has been much less successful. Since the free surface model has provided better results than the enriched pressure two phase model in the monolithic case, the combination with the velocity correction can be considered a priority, specially if bigger problems than the ones presented in this thesis must be solved.

Our free surface model does not take air into account. As we have mentioned in the conclusions of Chapter 5 modifications such as those presented in [77] or [16] can be introduced to calculate the pressure in air bubbles. This pressure is then applied as a normal traction at the free surface to take into account the effect of air. In flows where the air is compressed in its interaction with the fluid this model would provide a more accurate representation than a two phase flow incompressible model. The implementation of this modification in our code could extend the range of problems we can solve.

The solution of interface flows on fixed has two basic steps. This thesis has focused on problems relating the solution of the Navier Stokes equations. Now that sufficient progress has been made in this area it seems logical that improvements should also be introduced in the transport of the Level Set equation. The key point is the reinitialization of the Level Set function so that it remains smooth (close to a signed distance function). During the reinitialization process the interface displacement must be minimized, that is, mass must be locally conserved. A fresh approach that could easily be extended to unstructured meshes has been presented in [55].

Two alternatives for the solution of the Navier Stokes equations have been used in this thesis: the straightforward solution of the monolithic system and pressure segregation methods. Usually CFD groups stick to one or the other approach. Closing the gap between the two alternatives is an interesting line of research.

Bibliography

[1] P. Amestoy, I. Duff, J. Koster, and J.-Y. L'Excellent. A fully asynchronous multifrontal solver using distributed dynamic scheduling. *SIAM Journal of Matrix Analysis and Applications*, 23:15–41, 2001.

[2] P. Amestoy, I. Duff, and J.-Y. L'Excellent. Multifrontal parallel distributed symmetric and unsymmetric solvers. *Computer Methods in Applied Mechanics and Engineering*, 184:501–520, 2000.

[3] R. Aubry. *Three dimensional Lagrangian fluid flow with thermal coupling.* PhD thesis, Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports, Universitat Politècnica de Catalunya, Barcelona, 2006.

[4] S. Badia. *Stabilized Pressure Segregation Methods and their Application to Fluid-Structure Interaction Problems.* PhD thesis, Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports, Universitat Politècnica de Catalunya, Barcelona, 2005.

[5] S. Badia and R. Codina. Algebaric pressure segregation methods for the incompressible Navier-Stokes equations. *Arch Comput Methods Eng*, 15:343–369, 2008.

[6] S. Badia and R. Codina. Pressure segregation methods based on a discrete pressure Poisson equation. an algebraic approach. *International Journal for Numerical Methods in Fluids*, 56:351–382, 2008.

[7] C. Baiocchi, F. Brezzi, and L. Franca. Virtual bubbles and Galerkin/least-squares type methods (Ga.L.S). *Computer Methods in Applied Mechanics and Engineering*, 105:125–141, 1993.

[8] G. Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, 1967.

[9] M. Behr. On the application of slip boundary conditions on curved boundaries. *International Journal for Numerical Methods in Fluids*, 45:43–51, 2004.

[10] J. Bell and D. Marcus. A second-order projection method for variable-density flows. *JCP*, 101:334–348, 1992.

[11] J. Blasco, R. Codina, and A. Huerta. A fractional step method for the incompressible Navier-Stokes equations related to a predictor-multicorrector algorithm. *International Journal for Numerical Methods in Fluids*, 28:1391–1419, 1998.

[12] F. Brezzi and M. Fortin. *Mixed and hybrid finite element methods*. Springer Verlag, 1991.

[13] F. Brezzi, L. Franca, T. Hughes, and A. Russo. $b = \int g$. *Computer Methods in Applied Mechanics and Engineering*, 145:329–339, 1997.

[14] A. Brooks and T. Hughes. Streamline upwind / Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equation. *Computer Methods in Applied Mechanics and Engineering*, 32:199–259, 1982.

[15] D. Brown, R. Cortez, and M. Minion. Accurate projection methods for the incompressible Navier-Stokes equations. *JCP*, 168:464–499, 2001.

[16] A. Caboussat. Numerical simulation of two–phase free surface flows. *Archives of Computational Methods in Engineering*, 12:165–224, 2005.

[17] J. Cahouet and J. Chabard. Some fast 3d finite element solvers for the generalized Stokes problem. *IJNMF*, 8:869–895, 1988.

[18] Y. Chang, T. Hou, B. Merriman, and S. Osher. A level set formulation of Eulerian interface capturing methods. *Journal of Computational Physics*, 124:449–464, 1996.

[19] J. Chessa and T. Belytschko. A extended finite element method for two–phase fluids. *Journal of Applied Mechanics*, 70:10–17, 2003.

[20] A. Chorin. A numerical method for solving incompressible viscous problems. *Journal of Computational Physics*, 2:12–26, 1967.

[21] R. Codina. *A Finite Element Model for Incompressible Flow Problems.* PhD thesis, Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports, Universitat Politècnica de Catalunya, Barcelona, 1992.

[22] R. Codina. A discontinuity-capturing crosswind-dissipation for the finite element solution of the convection-diffusion equation. *Computer Methods in Applied Mechanics and Engineering*, 110:325–342, 1993.

[23] R. Codina. Stabilization of incompressibility and convection through orthogonal sub-scales in finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 190:1579–1599, 2000.

[24] R. Codina. A stabilized finite element method for generalized stationary incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 190:2681–2706, 2001.

[25] R. Codina. Stabilized finite element approximation of transient incompressible flows using orthogonal subscales. *Computer Methods in Applied Mechanics and Engineering*, 191:4295–4321, 2002.

[26] R. Codina and S. Badia. On some pressure segregation methods of fractional-step type for the finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 195:2900–2918, 2006.

[27] R. Codina and J. Baiges. Approximate imposition of boundary conditions in immersed boundary methods. *International Journal for Numerical Methods in Engineering*, Submitted.

[28] R. Codina, A. Coppola-Owen, P. Nithiarasu, and C.Liu. Numerical comparison of CBS and SGS as stabilization techniques for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Engineering*, 66:1672–1689, 2006.

[29] R. Codina, G. Houzeaux, H. Coppola-Owen, and J. Baiges. The fixed-mesh ALE approach for the numerical approximation of flows in moving domains. *Journal of Computational Physics*, 228:1591–1611, 2009.

[30] R. Codina, J. Principe, O. Guasch, and S. Badia. Time dependent subscales in the stabilized finite element approximation of incompressible flow problems. *Computer Methods in Applied Mechanics and Engineering*, 196:2413–2430, 2007.

[31] R. Codina, U. Schäfer, and E. Oñate. Mould filling simulation using finite elements. *International Journal of Numerical Methods for Heat & and Fluid Flow*, 4:291–310, 1994.

[32] R. Codina and O. Soto. A numerical model to track two-fluid interfaces based on a stabilized finite element method and the level set technique. *International Journal for Numerical Methods in Fluids*, 40:293–301, 2002.

[33] R. Codina and O. Soto. Approximation of the incompressible Navier–Stokes equations using orthogonal–subscale stabilization and pressure segregation on anisotropic finite element meshes. *Computer Methods in Applied Mechanics and Engineering*, 193:1403–1419, 2004.

[34] R. Codina, M. Vázquez, and O. Zienkiewicz. A general algorithm for compressible and incompressible flow—Part III. The semi-implicit form. *International Journal for Numerical Methods in Fluids*, 27:13–32, 1998.

[35] S. E. R. I. Community. http://wiki.manchester.ac.uk/spheric/index.php.

[36] A. Coppola-Owen and R. Codina. An improved level-set approach using finite elements with discontinuous gradient pressure shape functions. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *International Conference on Computational Methods in Marine Engineering MARINE 2005*, pages 463–477. CIMNE, 2005.

[37] A. Coppola-Owen and R. Codina. Improving Eulerian two-phase flow finite element approximation with discontinuous gradient pressure shape functions. *International Journal for Numerical Methods in Fluids*, 49:1278–1304, 2005.

[38] A. Coppola-Owen and R. Codina. A finite element model for free surface flows on fixed meshes. *International Journal for Numerical Methods in Fluids*, 54:1151–1171, 2007.

[39] H. Coppola-Owen and R. Codina. Free surface flows on fixed meshes. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *International Conference on Computational Methods in Marine Engineering MARINE 2007*, pages 463–477. CIMNE, 2007.

[40] M. Cross, K. Pericleous, T. Croft, D. McBride, J. Lawrence, and A. Williams. Computational modeling of mold filling and related free-surface flows in shape casting: An overview of the challenges involved. *Metallurgical and Materials Transactions B*, 37B:879–885, 2006.

[41] R. Elias and A. Coutinho. Stabilized edge-based finite element simulation of free-surface flows. *International Journal for Numerical Methods in Fluids*, 54:965–993, 2007.

[42] H. Elman. Preconditioning strategies for models of incompressible flow. *Journal of Scientific Computing*, 25:347–366, 2005.

[43] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers*. Oxford University Press, 2005.

[44] M. Engelman, V. Hartounian, and I. Hasbani. Segregated finite element algorithms for the numerical solution of large-scale incompressible flow problems. *International Journal for Numerical Methods in Fluids*, 17:323–348, 1993.

[45] T. P. Fries. The intrinsic XFEM for two-fluid flows. *International Journal for Numerical Methods in Fluids*, 60:437–471, 2009.

[46] D. Gao. A three dimensional finite element-volume tracking model for mould filling in casting processes. *International Journal for Numerical Methods in Fluids*, 29:877–895, 1999.

[47] J. Garcia-Espinosa, A. Valls, and E. Oñate. ODDLS: A new unstructured mesh finite element method for the analysis of free surface flow problems. *International Journal for Numerical Methods in Engineering*, 76:1297–1327, 2008.

[48] P. M. Gresho and R. Sani. *Incompressible flow and the finite element method*. John Wiley & Sons, 2000.

[49] O. Guasch and R. Codina. A heuristic argument for the sole use of numerical stabilization with no physical LES modeling in the simulation of incompressible turbulent flows. *Journal of Computational Physics*, submitted.

[50] J. Guermond, P. Minev, and J. Shen. An overview of projection methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, 195:6011–6045, 2006.

[51] J. Guermond and L. Quartapelle. A projection fem for variable density incompressible flows. *JCP*, 165:167–188, 2000.

[52] J. Guermond and J. Shen. Velocity-correction projection method for incompressible flows. *SIAM Journal on Numerical Analysis*, 41:112–134, 2003.

[53] A. Hansbo and P. Hansbo. An unfitted finite element method, based on Nitsche's method, for elliptic interface problems. *Computer Methods in Applied Mechanics and Engineering*, 191:5537–5552, 2002.

[54] F. Harlow and J. Welch. Numerical study of large-amplitude free-surface motions. *Physics of Fluids*, 9:842–851, 1966.

[55] D. Hartmann, M. Meinke, and W. Schröder. Differential equation based constrained reinitialization for level set methods. *Journal of Computational Physics*, 227:6821–6845, 2007.

[56] J. Heywood and R. Rannacher. Finite element approximation of the nonstationary Navier-Stokes problem. IV: Error analysis for second-order time discretization. *SIAM Journal on Numerical Analysis*, 27:353–384, 1990.

[57] C. Hirt and B. Nichols. Volume of fluid (VOF) method for the dynamics of free boundaries. *Journal of Computational Physics*, 39:201–225, 1981.

[58] G. Houzeaux and R. Codina. Transmission conditions with constraints in finite element domain decomposition methods for flow problems. *Communications in Numerical Methods in Engineering*, 17:179–190, 2001.

[59] G. Houzeaux and R. Codina. A finite element model for the simulation of lost foam casting. *International Journal for Numerical Methods in Fluids*, 46:203–226, 2004.

[60] T. Hughes. *The Finite Element Method*. Prentice-Hall, 1987.

[61] T. Hughes. Multiscale phenomena: Green's function, the Dirichlet-to-Neumann formulation, subgrid scale models, bubbles and the origins of stabilized formulations. *Computer Methods in Applied Mechanics and Engineering*, 127:387–401, 1995.

[62] T. Hughes, L. Franca, and M. Balestra. A new finite element formulation for computational fluid dynamics: V. Circumventing the Babuška-Brezzi condition: a stable Petrov-Galerkin formulation for the Stokes problem accommodating equal-order interpolations. *Computer Methods in Applied Mechanics and Engineering*, 59:85–99, 1986.

[63] T. Hughes, L. Mazzei, and K. Jansen. Large eddy simulation and the variational multiscale method. *Computing and Visualization in Science*, 3:47–59, 2000.

[64] S. Idelsohn, E. Oñate, F. D. Pin, and N. Calvo. Fluid-structure interaction using the particle finite element method. *Computer Methods in Applied Mechanics and Engineering*, 195:2100–2123, 2006.

[65] K. Jansen, S. Collis, C. Whiting, and F. Shakib. A better consistency for low-order stabilized finite element methods. *Computer Methods in Applied Mechanics and Engineering*, 174:153–170, 1999.

[66] C. Johnson. *Numerical Solution of Partial Differential Equations by the Finite Element Method*. Cambridge University Press, 1987.

[67] G. Karniadakis, M. Israeli, and S. Orzag. High order splitting methods for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 59:414–443, 1991.

[68] D. Kay and D. Loghin. A Green's function preconditioner for steady state Navier-Stokes equations. Technical Report NA-99/06, Oxford University Computing Lab, 1999.

[69] K. Kleefsman, G. Fekken, A. Veldman, B. Iwanowski, and B. Buchner. A volume-of-fluid based simulation method for wave impact problems. *Journal of Computational Physics*, 206:363–393, 2005.

[70] D. Kothe. Perspective on eulerian finite volume methods for incompressible interfacial flows. In H. Kuhlmann and H. Rath, editors, *Free Surface Flows*, pages 267–331. Springer-Verlag, 1999.

[71] A. Larese, R. Rossi, E. Oñate, and S. Idelsohn. Validation of the particle finite element method (PFEM), for simulation of free surface flows. *Engineering Computations*, 25:385–425, 2008.

[72] B. E. Launder and D. B. Spalding. The numerical computation of turbulent flows. *Computer Methods in Applied Mechanics and Engineering*, 3:269–289, 1974.

[73] R. Lewis, A. Usmani, and J. Cross. Efficient mould filling simulation in metal castings by an explicit finite element method. *International Journal for Numerical Methods in Engineering*, 20:493–506, 1995.

[74] D. Loghin and A. Wathen. Schur complement preconditioners for the Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 40:403–412, 2002.

[75] R. Löhner, C. Yang, J. Cebral, F. Camelli, O. Soto, and J. Waltz. Improving the speed and accuracy of projection-type incompressible flow solvers. *CMAME*, 195:3087–3109, 2006.

[76] R. Löhner, C. Yang, and E. Oñate. Large-scale simulation of flows with violent free surface motion. In P. Bergan, J. García, E. Oñate, and T. Kvamsdal, editors, *International Conference on Computational Methods in Marine Engineering MARINE 2005*, pages 55–81. CIMNE, 2005.

[77] R. Löhner, C. Yang, and E. Oñate. On the simulation of flows with violent free surface motion. *CMAME*, 195:5597–5620, 2006.

[78] R. Löhner, C. Yang, E. Oñate, and S. Idelsohn. An unstructured grid-based, parallel free surface solver. *Applied Numerical Mathematics*, 31:271–293, 1999.

[79] J. López and J. Hernández. Analytical and geometrical tools for 3D volume of fluid methods in general grids. *Journal of Computational Physics*, 227:5939–5948, 2008.

[80] V. Maronnier, M. Picasso, and J. Rappaz. Numerical simulation of three-dimensional free surface flows. *International Journal for Numerical Methods in Fluids*, 42:696–716, 2003.

[81] P. Minev, T.Chen, and K.Nandakumar. A finite element technique for multfluid incompressible flow using Eulerian grids. *Journal of Computational Physics*, 187:255–273, 2003.

[82] F. Mut. Algunas contribuciones al level set method en elementos finitos estabilizados. Master's thesis, Instituto Balseiro, Universidad Nacional de Cuyo, 2003.

[83] F. Mut, G. Buscaglia, and E. Dari. New mass-conserving algorithm for level set redistancing on unstructured meshes. *Journal of Applied Mechanics*, 73:1011–1016, 2006.

[84] N. Nikitin. Third-order-accurate semi-implicit Runge-Kutta scheme for incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluids*, 51:221–233, 2006.

[85] J. Oden and L. Demkowicz. *Applied Functional Analysis*. CRC Press, 1996.

[86] E. Oñate. A stabilized finite element method for incompressible viscous flows using a finite increment calculus formulation. *Computer Methods in Applied Mechanics and Engineering*, 182:355–370, 2000.

[87] E. Oñate, S. Idelsohn, F. D. Pin, and R. Aubry. The particle finite element method. an overview. *International Journal of Computational Methods*, 1:267–307, 2004.

[88] E. Oñate, S. Idelsohn, and C. Sacco. Finite element solution of free surface ship wave problems. *International Journal for Numerical Methods in Engineering*, 45:503–528, 1999.

[89] S. Osher and R. Fedkiw. Level set methods: and overview and some recent results. *Journal of Computational Physics*, 169:463–502, 2001.

[90] S. Osher and R. Fedkiw. *Level set methods and dynamic implicit surfaces*. Springer–Verlag, 2003.

[91] S. Osher and J. Sethian. Fronts propagating with curvature dependent speed: algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79:12–49, 1988.

[92] T. Papanastasiou, N. Malamataris, and K. Ellwood. A new outflow boundary condition. *International Journal for Numerical Methods in Fluids*, 14:587–608, 1992.

[93] J. Perot. An analysis of the fractional step method. *Journal of Computational Physics*, 108:51–58, 1993.

[94] E. Pichelin and T. Coupez. Finite element solution of the 3D mold filling problem for viscous incompressible fluid. *Computer Methods in Applied Mechanics and Engineering*, pages 359–371, 163.

[95] J. Principe, R. Codina, and F. Henke. The dissipative structure of variational multiscale methods for incompressible flows. *Computer Methods in Applied Mechanics and Engineering*, accepted.

[96] A. Quarteroni, F. Saleri, and A. Veneziani. Factorization methods for the numerical approximation of Navier-Stokes equations. *Computer Methods in Applied Mechanics and Engineering*, 188:505–526, 2000.

[97] M. Quecedo and M. Pastor. Application of the level set method to the finite element solution of two-phase flows. *International Journal for Numerical Methods in Engineering*, 50:645–663, 2001.

[98] R. Radovitzky and M. Ortiz. Lagrangian finite element analysis of Newtonian fluid flows. *Journal of Computational Physics*, 43:607–619, 1998.

[99] B. Ramaswamy. Numerical simulation of unsteady viscous free surface flow. *Journal of Computational Physics*, 90:396–430, 1990.

[100] B. Ramaswamy, M. Kawahara, and N. T. Lagrangian finite element method for the analysis of two-dimensional sloshing problems. *International Journal for Numerical Methods in Fluids*, 6:659–670, 1985.

[101] G. Ren and T. Utnes. A finite element solution of the time-dependent incompressible Navier-Stokes equations using a modified velocity correction method. *International Journal for Numerical Methods in Fluids*, 17:349–364, 1993.

[102] Y. Saad. *Iterative methods for sparse linear systems.* PWS, 1996.

[103] R. Scardovelli and S. Zaleski. Direct numerical simulation of free-surface and interfacial flow. *Annu. Rev. Fluid Mech.*, 31:567–603, 1999.

[104] J. Sethian and P. Smereka. Level set methods for fluid interfaces. *Annu. Rev. Fluid Mech.*, 35:341–372, 2003.

[105] O. Soto, R. Löhner, J. Cebral, and R. Codina. A time–accurate implicit–monolithic finite element scheme for incompressible flow problems. In *Eccomas CFD 2001,* CD proceedings, 2001.

[106] M. Sussman, A. Almgren, J. Colella, L. Howell, and M. Welcome. An adaptive level set approach for incompressible two phase flows. *Journal of Computational Physics*, 148:81–124, 1999.

[107] R. Temam. Sur l'approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionaires (I). *Archives for Rational Mechanics and Analysis*, 32:135–153, 1969.

[108] T. Tezduyar. Stabilized finite element formulations for incompressible flow computations. *Advances in Applied Mechanics*, 28:1–44, 1991.

[109] T. Tezduyar. Interface-tracking, interface-capturing and enhanced solution techniques. In *Proceedings of the First South-American Congress on Computational Mechanics, Santa Fe - Parana, Argentina*, 2002.

[110] E. Thompson. Use of the pseudo-concentration to follow creeping viscous during transient analysis. *International Journal for Numerical Methods in Engineering*, 6:749–761, 1986.

[111] L. Timmermans, P. Minev, and F. V. de Vosse. An approximate projection scheme for incompressible flow using espectral elements. *International Journal for Numerical Methods in Fluids*, 22:673–688, 1996.

[112] G. Tryggvason, B. Bunner, A. Esmaeeli, D. Juric, N. Al-Rawahi, W. Tauber, J. Han, S. Nas, and Y.-J. Janz. A front-tracking method for the computations of multiphase flow. *Journal of Computational Physics*, 169:708–759, 2001.

[113] S. Turek. A comparative study of time-stepping techniques for the incompressible Navier-Stokes equations: from fully implicit nonlinear schemes to semi-implicit projection methods. *International Journal for Numerical Methods in Fluids*, 22:987–1011, 1996.

[114] S. Turek. On discrete projection methods for thr incompressible Navier Stokes equations: An algorithmical approach. *Computer Methods in Applied Mechanics and Engineering*, 143:271–288, 1997.

[115] S. Turek. *Eficient Solvers for Incompressible Flow Problems*. Springer-Verlag, 1999.

[116] S. Van der Pijl. Free boundary methods for multi-phase flows. Technical Report 02-13, Delft University of Technology, 2002.

[117] J. van Kan. A second-order accurate pressure correction scheme for viscous incompressible flow. *SIAM Journal of Scientific Computing*, 7:870–891, 1986.

[118] Vulcan. http://www.quantech.es/quantechatz/vulcan.html.

[119] O. Zienkiewicz and R. Codina. A general algorithm for compressible and incompressible flow—Part I. The split, characteristic-based scheme. *International Journal for Numerical Methods in Fluids*, 20:869–885, 1995.