UNIVERSITÉ PARIS-EST
École Doctorale MODES

Thèse de Doctorat

Domaine:
Génie-Civil

Dissertation zur Erlangung
des akademischen Grades
Doktoringenieur (Dr.-Ing.)

Wissenschaftlichesrechnen

Christophe Kassiotis

Nonlinear Fluid-Structure Interaction: a Partitioned Approach and its Application Through Component Technology

Defense : November 20^{th}, 2009.

Jury		
J.-B. Colliat	Encadrant	ÉNS-Cachan
D. Duhamel	Directeur de Thèse	École des Ponts ParisTech
J.-M. Ghidaglia	Examinateur	ÉNS-Cachan
A. Ibrahimbegović	Directeur de Thèse	ÉNS-Cachan
P. Massin	Examinateur	ÉDF R&D
H. G. Matthies	Directeur de Thèse	TU-Braunschweig
D. Perić	Rapporteur	Swansea University
F.-X. Roux	Rapporteur	Onera/UPMC

UR-Navier — École des Ponts - ParisTech
Institüt für Wissenschaftliches Rechen — TU–Braunschweig
LMT-Cachan — École Normale Supérieur de Cachan

À mon grand-père, Vasilli Kassiotis.

"Science is what we understand well enough to explain to a computer.
Art is everything else we do."

— D. E. Knuth, 1996.

Résumé

Interaction fluide-structure non-linéaire: une approche partitionnée et son application par la technologie des composants

Au cours de ces travaux de thèse, la résolution de problèmes non-linéaires en interaction forte entre une structure et un fluide a été étudiée par une approche partitionnée. La stabilité, la convergence et les performances de différents schémas de couplages explicites et implicites ont été explorées.

L'approche partitionnée autorise la réutilisation des codes existants dans un contexte plus général. Un des objectifs de nos travaux est de les utiliser comme des boites noires, dont on n'a pas le besoin de connaitre le fonctionnement interne. A cette fin, la technologie des composants et le *middleware* CTL ont été utilisés. Ainsi, deux composants basés sur des codes existants pour le fluide et la structure ont été développés puis couplés par une approche de type code maître. Les performances de différentes architectures de composants aussi bien que la communication entre composants parallélisés sont décrites dans ce document.

La réutilisation de codes existants permet de profiter au plus vite des modèles avancés développés de manière spécifique pour nos sous-problèmes. Pour la partie solide, par exemple, il est possible d'utiliser différents modèles Éléments Finis en grandes déformations avec des matériaux non-linéaires. Pour la partie fluide, nous avons choisi une approche Arbitrairement Lagrangienne-Eulérienne, et la résolution par Volumes Finis. Différents régimes d'écoulements instationnaires – aussi bien incompressibles (modélisés alors par les équations de Navier-Stokes) qu'à surfaces libres – sont ici considérés. La description de phénomènes tels que le déferlement des vagues et leur impact sur des structures est ainsi rendu possible.

Mots-clefs: Interaction fluide-structure, couplage fort, approche partitionnée, couplage de codes, technologie des composants.

Zusammenfassung

Nichtlineare Fluid-Struktur-Interaktion: Ein partitionierter Ansatz und dessen softwarekomponentenbasierte Umsetzung

In dieser Doktorarbeit wird ein partiotionierter Ansatz zur Lösung nichlinearer stark gekoppelter Fluid-Struktur-Interaktionsprobleme behandelt. Dabei werden die Stabilität, die Konvergenz und die Performanz expliziter und impliziter Kopplungsalgorithmen untersucht.

Der partitionierte Ansatz ermöglicht die Wiederverwendung von existierender Software in einem allgemeineren Kontext. Ein Ziel dieser Arbeit ist hierbei die Nutzung dieser Software als Blackboxen. Hierzu verwenden wir das komponentenbasierte Framework CTL. Die existierenden Simulationscodes für das Strömungs- und das Strukturproblem werden als CTL Komponenten umgesetzt und über einen Mastercode gekoppelt. Die Performanz des Gesamtsystems wird hinsichtlich unterschiedlicher Komponentenbindungen und der parallelen Implementierungen der Simulationskomponenten analysiert.

Existierende Simulationscodes weisen mitunter viele Mannjahre Entwicklungszeit auf, bieten auf die einzelnen Probleme abgestimmte numerische Verfahren und unterstützen unterschiedliche Modelle des betrachteten physikalischen Fachgebietes. Daher ist eine Wiederverwendung erstrebenswert. Der Strukturteil wird über die Finite Elemente Methode approximiert, wobei große Deformationen und verschiedene nicht-lineare Materialmodelle unterstützt werden. Auf der Strömungsseite werden Beispielprobleme (von instationären inkompressiblen Strömungen zu Strömungen mit freier Oberfläche) herangezogen, die mit der Arbitrary Lagrangian Eulerian Methode formuliert und der Finite Volumen Methode diskretisiert werden.

Schlagworte: Fluid-Struktur-Interaktion, starke Kopplung, partitionierter Ansatz, Softwarekopplung, Komponententechnologie.

Abstract

Nonlinear Fluid-Structure Interaction: a Partitioned Approach and its Application Through Component Technology

A partitioned approach is studied to solve strongly coupled nonlinear fluid-structure interaction problems. The stability, convergence and performance of explicit and implicit coupling algorithms are explored.

The partitioned approach allows to re-use existing codes in a more general context. One purpose of this work is to be able to couple them as black-boxes. To that end, the scientific software component framework CTL is considered. Therefore a fluid and a structure component based on existing software are developed and coupled with a master code approach. Computational performance of different remote calls and parallel implementation of components are also depicted herein.

The re-use of existing software allows to couple advanced models developed for both sub-problems. In this work, the structure part is solved by the Finite Element Method, with the possibility to use different non-linear and large deformation behaviors. For the fluid part, examples modeled with an Arbitrary Lagrangian Eulerian formulation are considered, solved with a Finite Volume Method. The models used are first transient incompressible flows described by the Navier-Stokes equation, then free surface flows. With the latter, the impact of sloshing and breaking waves on model structures can be computed

Keywords: Fluid-structure interaction, strong coupling, partitioned approach, software coupling, component technology.

Acknowledgments – Remerciements – Danke

Je commencerai ces traditionnels remerciements par mes directeurs de thèse Adnan Ibrahimbegović et Hermann Matthies qui m'ont permis de travailler dans les meilleures conditions possibles : un encadrement scientifique de haut niveau, des approches complémentaires des problèmes abordés, la possibilité de réaliser cette thèse en cotutelle avec l'Allemagne et surtout leur passion bien réelle pour la recherche. Je tiens aussi à remercier les responsables du Corps des Ponts de m'avoir permis de continuer mes travaux de thèse déjà entamés, et par la même occasion Denis Duhamel d'avoir accepté de m'encadrer.

Je suis extrêmement fier d'avoir eu des rapporteurs de la qualité de Djordje Perić et François-Xavier Roux, qui ont accepté de lire attentivement cette thèse (bien qu'écrite dans mon anglais parfois approximatif). Je remercie aussi Patrick Massin d'avoir accepté de faire partie de mon jury, ainsi que Jean-Michel Ghidaglia de m'avoir fait l'honneur de présider ce même jury.

Avoir fait une thèse en cotutelle m'oblige ici à massacrer la langue de Goethe. Ich danke meinem deutschen Kollegen für ihre unschätzbare Hilfe während meines Aufenthalts in Braunschweig, und unter ihnen: Martin Krosche (for conversations in english and "broken german", for office sharing and for the horse), Rainer Niekamp (CTL master, so kind and helpfull with my `C++` questions), Cosima Meyer (for her kindness and help with all the administrative tasks), Dominik Jürgens und Oliver Pajonk ...

Je voudrais aussi remercier tous mes collègues du LMT-Cachan et amis de France pour les trois années (modulo mes séjours en Allemagne) que j'ai passées avec eux, et notamment, par ordre alphabétique, pour ne pas faire de jaloux, et même si vous rêvez tous de connaître votre CK-index qui restera donc secret : Carla Austruy (ma stagiaire préférée), Nathan Benkemoun (pour ses petites poutres et m'avoir fait rêver à mon anniversaire), Benoit Blaysat, Amor Boulkertous, Pierre-Étienne Charbonnel (pour *Poster Girl*), Jean-Baptiste Colliat (pour m'avoir initié aux composants en Master 2), Renaud Costadoat, Flavien Fremy (pour les résumé de l'Équipe au retour de week-end), Martin Genet (pour le Journal Club où j'étais le troisième Martin), Louis Kovalesky, Martin Hautefeuille (pour tout ce que nous avons programmé ensemble et bien plus), Thomas de Larrard (pour ses lasagnes au cardon), Roxane Marull (pour sa rigueur typo- et orthographique), Ayman Moussa (pour la relation formulation faible/volume finis), Elena Ourjoumt-

seva, Florent Pled (pour son enthousiasme), Lavinia Stefan (pour les armes de poing et le vin roumain), Vincent Thomas, Florian Thomines (camarade), et Julien Waytens (pour le foot du mercredi). Si vous n'êtes pas un copain du foot, un collègue, un élève de l'ENS-Cachan ou une des nombreuses autres personnes à laquelle je pense en écrivant ces lignes tout en voulant éviter de dépasser deux pages, vous survivrez à mon étourderie.

J'aimerais remercier profondément ma famille, mes sœurs, mon frère et mes parents d'avoir supporté mes longues absences, mon discours abscons de 45 minutes le jour de ma soutenance et d'avoir organisé un pot si extraordinaire.

Enfin, je tiens à exprimer ma reconnaissance toute particulière à Julie Murat pour son soutien constant durant la période de rédaction et de préparation à la soutenance, les corrections attentives qu'elle a pu apporter à mon anglais douteux, et tout ce que je ne saurais mettre dans ces remerciements de thèse.

Chatou, Novembre 2009.

Contents

Introduction **1**

1 Structure and fluid subproblems **5**
1.1 Solving structure problems with FEM 8
1.1.1 Strong form of the structure problem 8
1.1.2 Weak forms of the structure problem and Finite Element application 9
1.1.3 Time integration of a discretized Finite Element problem 10
1.2 Incompressible flows solved by FVM 11
1.2.1 Strong form of the Navier-Stokes equations 11
1.2.2 Navier-Stockes equations discretization 12
1.2.3 Pressure-Implicit with Splitting of Operators (PISO) algorithm 15
1.2.4 Validating the CFD strategy 16
1.2.5 Two-dimensional numerical experiments 17
1.2.6 Three-dimensional numerical experiments 20
1.3 Flow in a moving shape domain 21
1.3.1 Arbitrary Lagrangian-Eulerian strategy 23
1.3.2 Free-surface flows 26
Closure 30

2 Partitioned Approach for FSI **33**
2.1 Solving a coupled problem 36
2.1.1 Monolithic strategy 36
2.1.2 Partitioned strategy 37
2.1.3 Partitioned strategy notations 38
2.2 Explicit coupling 41
2.2.1 Generalized Conventional Serial Staggered algorithm . . . 41
2.2.2 Evaluation of interface energy for DFMT-GCSS 43
2.2.3 Enforcement of the Geometric Conservation Law 45
2.2.4 Improved Serial Staggered (ISS) algorithm 45
2.2.5 Conventional Parallel Staggered (CPS) algorithm 47
2.2.6 Explicit coupling with incompressible flows: the artificial "Added-Mass Effect" 47
2.3 Implicit coupling strategies 48
2.3.1 Algebraic solvers based on Picard iterations 49

2.3.2 Relaxation techniques . . . 51
2.3.3 Newton and quasi-Newton based strategy . . . 55
Closure . . . 58

3 Components for FSI **61**
3.1 Component technology framework . . . 64
3.1.1 Component Oriented Programming paradigm . . . 64
3.1.2 Component-based implementation and its specifics . . . 65
3.1.3 The middleware CTL . . . 65
3.2 Structure component based on `FEAP` . . . 66
3.2.1 Interface definition of the structure component . . . 67
3.2.2 Implementation of coFeap . . . 68
3.2.3 Calling a service from the mechanical component . . . 70
3.3 Fluid component based on `OpenFOAM` . . . 71
3.3.1 Component interface and its implementation . . . 71
3.3.2 RPC versus system calls and file reading performances . . . 75
3.3.3 Parallel CFD Component Features . . . 77
3.4 The master code **co**ps . . . 87
3.4.1 Software coupling applied to fluid-structure interaction . . . 87
3.4.2 Component architecture of **co**ps . . . 88
3.4.3 Field interpolation between solvers . . . 88
Closure . . . 91

4 FSI numerical examples **93**
4.1 Driven cavity with flexible bottom . . . 95
4.1.1 The lid-driven cavity fluid problem . . . 95
4.1.2 Modification for the FSI validation case . . . 96
4.1.3 Subproblems discretization and numerical parameters . . . 99
4.1.4 Tight coupling with the modified lid-driven cavity case . . . 99
4.1.5 Implicit strong coupling and reference solution . . . 99
4.1.6 On the impossibility to apply explicit coupling . . . 101
4.2 Performances and fluid domain decomposition . . . 104
4.3 Flexible appendix in a flow . . . 106
4.3.1 Problem description . . . 106
4.3.2 Implicit coupling and reference solution . . . 109
4.3.3 Explicit coupling . . . 110
4.4 Three-dimensional flag in the wind . . . 112
4.4.1 Problem description . . . 112
4.4.2 Fluid discretization . . . 113
4.4.3 Solid discretization . . . 114
4.4.4 Coupling . . . 114
4.4.5 Results . . . 115
4.5 Two-dimensional wave hitting a structure . . . 116
4.5.1 Introduction and a first approach . . . 116
4.5.2 Problem description . . . 117
4.5.3 Fluid discretization . . . 117
4.5.4 Structure discretization . . . 118
4.5.5 Coupling . . . 118

4.5.6 Results 119
4.6 Three-dimensional wave impacting a structure 121
4.6.1 Problem description 121
4.6.2 Fluid discretization 121
4.6.3 Solid discretization 122
4.6.4 Coupling 122
4.6.5 Results 123
Closure 125

Conclusion **127**

A DFMT-BGS stability and convergence **129**
A.1 Reformulation of the FSI problem in DAE 129
A.2 Error propagation, stability and convergence 130
A.3 Proofs for stable error propagation 132

References **139**

Introduction

Contrary to the work of craftsmen which rely on pre-scientific rules based on trial and error, the power of engineers is to provide models in order to study, dimension and then build artifacts that will be used in a certain context. Therefore, it is necessary to take into account the effects of a lot – if not all – the environment physics on the studied object. However, one wants the computing of models to take only a limited amount of time. The first step is to simplify the studied object to a mechanical model – that often has to be simplified itself in order to be computed, either analytically or by numerical techniques. The multiphysics nature of the problem is taken into account by a given load – based on other simplified models. For instance, one approach widely used in Civil Engineering is to infer the loading from rules provided by regulators (for instance Eurocode [Comité Européen de Normalisation, 2009] in Europe). In such cases, one can find, for instance, a model force for the loading of a building by wind on the walls, snow on the roof, or fire inside.

This is the usual approach, even for complex models based, on numerical techniques such as the Finite Element Method. For instance, in the first publication on FEM [Clough, 1960], the author wants to compute a dam. The loading of water is there provided by a simple model. This was surely sufficient for the precision required, but it is clear that the structural displacement of the dam under the loading will influence the loading itself.

When it is required to compute more precisely the influence of at least one of several possible physical field by more sophisticated model, the problem is called *multiphysics*. When the influence of one of the subproblems is weak, the coupling can be one way: a computation of the first field gives a loading applied to the the second one. Neglecting the reaction of the second subproblem leads to *weak* or *tight* coupling. When the loading of the model by another physic is significantly and inseparably modified, the coupling is called *strong*.

This Ph. D. work is focused on the numerical modeling of *strong* coupled problems between fluids and structures. Interaction of fluids and structures is a key issue for a variety of physical systems. The range of examples that come to mind easily in the engineering domain is vast and covers all the scales: from bio- and medical- (where the scale of problems is often less than a millimeter) [Nobile, 2001, Deparis, 2004] to Civil Engineering (where it can be hundreds of meters) [Piperno, 1998, Frandsen, 2004].

As fluid and solid mechanics both belong to the general field of continuum mechanics, it is possible to formulate each of them problem in the same

framework [Germain and Muller, 1990]. However, the development of numerical strategies to solve intrinsically complex solid and fluid problems was mostly carried out by different research communities. This lead to efficient but often incompatible solvers.

So, the fluid-structure interaction problem is often approached with the intention to extend one of the methods traditionally applied to one of the subproblems to the whole coupled system. For example, one can cite Eulerian descriptions applied to solids [Mehl et al., 2008] or Lagrangian formulation applied to the fluid flow [Idelsohn et al., 2003]. If those strategies often yield interesting results and new points of view, they often necessitate tremendous software implementation and basic testing. Furthermore, as one starts from more or less "nothing" into a new domain, the time before reaching high-level model coupling is often too long. Finally, since engineers habits are often hard to change, and since their previous works, as imperfect as they may be, need to be conserved, those methods are restricted to the research area.

In this work, we propose a totally different approach that relies on partitioned strategies (which are often used for multiphysics problems) and their software realization based on code coupling. The work herein proposes to couple in a generic way a Finite Element Method code for the solid part and solver based on the Finite Volume Method for the fluid part. From the first stage, this allows to solve the complete Navier-Stokes equation in an Arbitrary Lagrangian Eulerian framework, and a geometrically non-linear description of the structures. As the coupled software provides different models it is also possible to model more complex flows (with turbulence for instance, or free surface that will be presented in this work) and solid materials like in plasticity or damaging for instance.

The main goal of the present work is to perform fluid-structure interaction computations with a focus on the possibility to use advanced model and on numerical performances in order to compute three dimensional problems. The aimed application is the representation of breaking wave interaction with an academic model of protection structure. To reach this goal, it is first necessary to choose the right models for the fluid and the structure, then to couple them with stable and efficient enough algorithm, as well as to develop an adapted software environment, and last to validate the proposed approach on numerical examples.

Thus, the outline of the this dissertation is as follows:

Chapter 1: This chapter introduces the notations and the strategy chosen for the fluid and the structure part. We choose here the traditional Lagrangian formulation solved by the Finite Element Method for the solid part [Ibrahimbegović, 2009]. The time integration is carried out by implicit schemes. For the fluid part, an Arbitrary Lagrangian Eulerian Method [Hirt et al., 1997], discretized by Finite Volume Method [Ferziger and Perić, 2002] is used to perform the computation of the flow on the moving domain determined by the fluid-structure interaction interface. This strategy is widely used for incompressible flows, but not familiar for people coming from the structure and Finite Elements word; therefore, the validation of the numerical fluid strategy on benchmark

problems [Schäfer and Turek, 1996] and the required extension to perform free-surface flows [Dutykh, 2008] are detailed in Chapter 1.

Chapter 2: Once the way to solve the fluid and the structure problems defined in Chapter 1, the way to perform strongly coupled fluid-structure interaction problems is explained. After a general introduction on solving strategies for multiphysics problems, their application to fluid-structure interaction are recalled. The limitation of explicit coupling when coupling incompressible flows [Causin et al., 2005] requires the introduction and the use of implicit coupling algorithms. For its simplicity, a fixed-point strategy is here chosen, and acceleration of the convergence is obtained using a relaxation technique such as Aitken's [Küttler and Wall, 2008]. The stability of this kind algorithm is conditional, and a proof of this limitation is made and detailed in Appendix A.

Chapter 3: The partitioned coupling strategy defined in the previous Chapter allows to re-use existing code in a more general context. Its implementation is performed in the component technology framework, using the middleware CTL [Niekamp, 2005b]. A master-slave approach is used. It requires the development of a coupling code (COupling COmponents by a Partitioned Strategy: **co**ps) that handles the data exchange and the communication between the solid and the fluid components. These two components are based on the re-use of existing codes. For the structure part, the FEM code `FEAP` programmed in Fortran is used to build a component. For the fluid part, the FVM code `OpenFOAM` programmed in `C++` is embedded in the component framework with ofoam. The cost of fluid computation justifies the development of a parallel version of the component. The way to handle the communication between the several instances of a parallel component are detailed herein.

Chapter 4: The development of components coupled with explicit and implicit algorithms allows to compute various kinds of fluid-structure interaction problems. The strategy is first validated on a simple bi-dimensional problem [Wall, 1999]. The necessity to use implicit coupling algorithm is emphasized with this example. Long term computation, such as for wind acting on structure are carried out. They are here two- and three-dimension with a large number of d-o-f. Finally, the interaction of free-surface flow – as in waves breaking – over structures is validated in two dimensions [Walhorn et al., 2005] and performed in three dimensions.

1 Description and validation of the structure and fluid subproblems

In this chapter the techniques used for both fluid and structure parts of the coupled problem are presented. A brief remainder of the equations for structural dynamic problems as well as their solution thanks to the Finite Element Method is followed by an introduction of the Finite Volume Method to solve incompressible flow problems associated with the fluid part. Some benchmarks of the proposed strategy are validated here. In the last section of this chapter methods used for fluid in moving domain are introduced. The moving domain is bounded either by the deformable structure or by a fluid free-surface.

Contents

1.1 Solving structure problems with FEM 8
1.1.1 Strong form of the structure problem 8
1.1.2 Weak forms of the structure problem and Finite Element application 9
1.1.3 Time integration of a discretized Finite Element problem 10
1.2 Incompressible flows solved by FVM 11
1.2.1 Strong form of the Navier-Stokes equations 11
1.2.2 Navier-Stockes equations discretization 12
1.2.3 Pressure-Implicit with Splitting of Operators (PISO) algorithm 15
1.2.4 Validating the CFD strategy 16
1.2.5 Two-dimensional numerical experiments 17
1.2.5.1 Problem description 17
1.2.5.2 Test case 2D-1 (steady) 18
1.2.5.3 Test case 2D-2 (unsteady) 18
1.2.5.4 Test case 2D-3 (unsteady) 19
1.2.6 Three-dimensional numerical experiments 20
1.3 Flow in a moving shape domain 21
1.3.1 Arbitrary Lagrangian-Eulerian strategy 23
1.3.1.1 Navier-Stokes equation in an ALE framework 23
1.3.1.2 Mesh motion equations 24
1.3.1.3 PISO algorithm for Navier-Stokes equation discretized by Finite Volume in a moving domain 25
1.3.2 Free-surface flows 26
1.3.2.1 Strategies for free-surface flows 26
1.3.2.2 A biphasic model for free surface flows . . . 27
1.3.2.3 Dam break example 29
Closure 30

This chapter is dedicated to the introduction of the numerical resolution of fluid flows and solid structures. This is an introductory Chapter, and the readers who are well aware of the resolution of these problems can go directly to the next Chapter that deals with the coupling of theses subproblems. These problems both belong to the general field of continuum mechanics [Salençon, 2005]. However, the descriptions generally used for each of the subproblems are different: for the structure part, it is natural to follow material point motion in a Lagrangian formulation, while an Eulerian formulation is often preferred for the fluid part. Naturally, there are bridges between this two formulations.

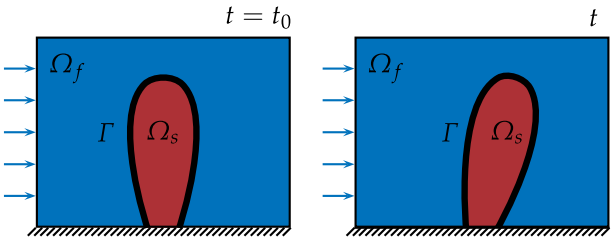


Figure 1.1: A fluid-structure interaction problem

For the solid part, the method widely used since the sixties is the Finite Element Method [Zienkiewicz and Taylor, 2001b, Ibrahimbegović, 2009], even if it is possible to solve the Partial Differential Equations that describe the continuum problem either by other discretization techniques such as Finite Volume [Slone et al., 2003]. In this work, a classic Finite Element Method is used to solve the this part of the coupled problem. Its description is given in Sec. 1.1.

For the fluid part, two different methods are mainly used: either the Finite Element Method [Zienkiewicz and Taylor, 2001a, Brooks and Hughes, 1990], or the Finite Volume Method [Ferziger and Perić, 2002]. The first one has the advantages of a strong mathematical framework and well established convergence properties, but requires special care to stabilize the solution when incompressible flow are aimed at [Franca et al., 1993]. For the second one, the mathematical framework is slightly more difficult to understand but the more physical foundation allows to naturally get good conservative properties. There are bridges that combine advantages of both formulations [Pascal and Ghidaglia, 2001].

In this work, the Finite Volume Method is chosen, since most of the commercial and non-commercial CFD solvers used in the industry for engineering problems rely on this technique. In Sec. 1.2 Validating a strategy used by a CFD solver is not an easy task. Among the promising tools are the goal-oriented error estimates, but most of them are developed for FEM-based strategies [Becker and Rannacher, 2003, Hoffman and Johnson, 2007], and really few are designed for the FVM based solver [Jasak, 1996, Bouche et al., 2006]. The chosen approach is hereby validated on benchmarks [Schäfer and Turek, 1996].

The last Section of this Chapter is dedicated to solving fluid problems in a moving domain. This is required when one aims at solving fluid-structure interaction problems, as the fluid domain shares at least one of its boundaries with the structure that is undergoing deformation. It is possible to solve this kind

of problem with for instance the fictitious domain approach [Wang et al., 2008] or with an Arbitrary Lagrangian-Eulerian approach [Demirdžić and Perić, 1988, Hirt et al., 1997]. The latter requires to solve the deformations of the fluid underlying grid, and is often not able to support too large deformation without re-meshing, but is often the one chosen in existing tools to solve fluid problems in moving domain, and was therefore used herein. The impossibility to represent too large deformations without re-meshing makes the choice of another method necessary when complex fluid domain deformations such as the ones observed for the sloshing of a wave are represented. Here a biphasic approach is chosen where both air and water are represented, and a characteristic function is used to distinguish the two phases.

1.1 Solving structure problems with the Finite Element Method

1.1.1 *Strong form of the structure problem*

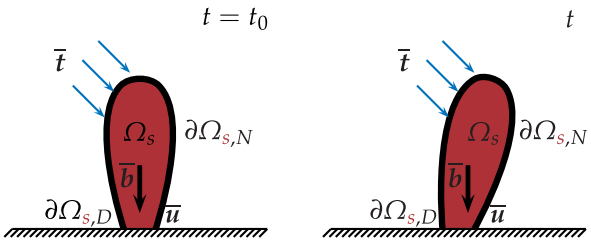


Figure 1.2: Solid problem

The structure is based on a Lagrangian description. Considering a structure domain Ω_s with imposed displacements $\overline{\boldsymbol{u}}$ on its Dirichlet boundary $\partial\Omega_{s,N}$ and under the loading of traction forces $\overline{\boldsymbol{t}}$ on its Neumann boundary $\partial\Omega_{s,N}$ as well as body load $\boldsymbol{b}$ on the whole domain Ω_s (see Fig. 1.2). The problem is dynamic and evolves in time on the segment $[0,T]$. The governing equation for a structure describes the momentum conservation and is also known as the Cauchy equation. The strong form of the structural problem written with respect to the deformed configuration is as follows:

Given: $\overline{\boldsymbol{u}}$ on $\partial\Omega_{s,D} \times [0,T]$, $\overline{\boldsymbol{t}}$ on $\partial\Omega_{s,N} \times [0,T]$ and $\overline{\boldsymbol{b}}$ in $\Omega_s \times [0,T]$.
Find: $\boldsymbol{u} \in \Omega_s \times [0,T]$ so that:

$$\nabla \cdot \boldsymbol{\sigma} + \rho_s \left(\overline{\boldsymbol{b}} - \partial_t^2 \boldsymbol{u} \right) = \boldsymbol{0} \quad \text{in } \Omega_s \times [0,T] \tag{1.1}$$

where ρ_s denotes the material density of the solid domain, $\boldsymbol{u}$ its displacement field, $\partial_t^2 \boldsymbol{u}$ the accelerations. The Cauchy stress tensor $\boldsymbol{\sigma}$ in the deformed configuration can be linked to the second Piola-Kirchhoff stress tensor $\mathbf{S}$ formulated in the initial configuration through the gradient of the deformation and its Jacobian.

To close this Partial Differential Equations system one needs to link the displacements (or one of its derived field) and the stresses together with behavior law:

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\mathbf{E}) \tag{1.2}$$

For instance, an elastic material model can be assumed to link the stress Cauchy tensor $\boldsymbol{\sigma}$ and the Green-Lagrange strain tensor $\mathbf{E}$:

$$\boldsymbol{\sigma} = \mathcal{C} : \mathbf{E} \tag{1.3}$$

where $\mathcal{C}$ denotes the constitutive fourth-order tensor. The non-linearity of the problem comes from the large displacement described by the following relation between Green-Lagrange tensor $\mathbf{E}$ and the material deformation gradient $\mathbf{F} = \nabla \boldsymbol{u}$:

$$\mathbf{E} = \frac{1}{2}\left(\mathbf{F}^T\mathbf{F} - \mathbf{I}\right) \tag{1.4}$$

The displacement field and the imposed forces are defined respectively in the Hilbert space $\mathcal{H}^1(\Omega_s)$ and Sobolev space $\mathcal{L}^2(\partial\Omega_{s,N})$ with dimensions $d = 2$ or 3. Therefore, the solution of the strong form problem is searched for in the following spaces:

$$\mathcal{U} = \left\{ \boldsymbol{u} \middle| \boldsymbol{u} \in \left[\mathcal{H}^1(\Omega_s)\right]^d, \boldsymbol{u}|_{\partial\Omega_{s,D}} = \overline{\boldsymbol{u}} \right\} \tag{1.5}$$

An associated vector space can be defined as such:

$$\mathcal{U}_0 = \left\{ \boldsymbol{u} \middle| \boldsymbol{u} \in \left[\mathcal{H}^1(\Omega_s)\right]^d, \boldsymbol{u}|_{\partial\Omega_{s,D}} = \boldsymbol{0} \right\} \tag{1.6}$$

1.1.2 *Weak forms of the structure problem and Finite Element application*

It is *a priori* impossible to find directly an exact solution in the solution space Eq. (1.5) to the problem defined in Eq. (1.1). The idea is to find the best approximation of the solution in a space where the solution can be found numerically. The FEM formulations [Zienkiewicz and Taylor, 2001c, Ibrahimbegović, 2006] derived from the equilibrium Eq. (1.1) rely on the associated weak forms of this problem:

Given: $\overline{\boldsymbol{t}}$ on $\partial\Omega_{s,N} \times [0,T]$ and $\overline{\boldsymbol{b}}$ in $\Omega_s \times [0,T]$.
Find: $\boldsymbol{u} \in \mathcal{U}$ such that,
For all: $\delta\boldsymbol{u} \in \mathcal{U}_0$

$$\begin{aligned} \mathcal{G}_s(\boldsymbol{u};\delta\boldsymbol{u}) &:= \int_{\Omega_s} \rho_s \partial_t^2 \boldsymbol{u} \cdot \delta\boldsymbol{u} + \int_{\Omega_s} \boldsymbol{\sigma} : \nabla\delta\boldsymbol{u} - \int_{\Omega_s} \overline{\boldsymbol{b}} \cdot \delta\boldsymbol{u} - \int_{\partial\Omega_s} \overline{\boldsymbol{t}} \cdot \delta\boldsymbol{u} \\ &= 0 \end{aligned}$$

The solid domain Ω_s is then discretized in a finite number of elements $\mathcal{T}_h = (\kappa_e)_{e=1,\dots,n_{el}}$ so that the whole space is covered by the finite elements that do not intersect. The research space associated with the solution is restrained to

the space of continuous piecewise polynomial functions on the Finite Elements. This space is noted:

$$\mathcal{U}^h = \mathcal{U} \cap \left\{ \boldsymbol{u} \in \mathcal{C}^0\left(\Omega_s\right) \middle| \boldsymbol{u}|_\kappa \in \mathcal{P}^p\left(\kappa\right), \forall \kappa \in \mathcal{T}_h \right\} \tag{1.7}$$

where $\mathcal{P}^p\left(\kappa\right)$ is the space of polynomials of order p on κ. The same restriction holds on the associated vector space. The FE problem is defined as:

Given: $\bar{\boldsymbol{t}}$ on $\partial\Omega_{s,N} \times [0,T]$ and $\bar{\boldsymbol{b}}$ in $\Omega_s \times [0,T]$.
Find: $\mathbf{u} \in \mathcal{U}^h$ such that,
For all: $\delta\mathbf{u} \in \mathcal{U}_0^h$

$$\mathcal{G}_s(\mathbf{u};\delta\mathbf{u}) = 0 \tag{1.8}$$

This semi-discrete problem can be written in a matrix form using the real valued vectors $\mathbf{u} \in \mathbb{R}^{n_{d-o-f}}$:

$$\mathbf{M}_s\ddot{\mathbf{u}} + \mathbf{K}_s\left(\mathbf{u}\right)\mathbf{u} = \mathbf{f} \tag{1.9}$$

with $\mathbf{M}$ the mass matrix, $\mathbf{K}$ the stifness matrix associated with a potentially non linear problem, and $\mathbf{f}$ the projected loading forces. They are properly defined by assembling locally computed matrix on each element with the polynomial basis N_e of $\mathcal{P}\left(\kappa_e\right)$:

$$\begin{aligned}
\mathbf{M}_{s,e} &= \int_{\kappa_e} \rho_s N_e^T N_e \, \mathrm{d}\Omega \\
\mathbf{K}_{s,e}(\mathbf{u}_e)\mathbf{u}_e &= \int_{\kappa_e} \nabla N : \boldsymbol{\sigma}\left(\mathbf{u}_e N_e\right) \, \mathrm{d}\Omega \\
\mathbf{f}_{s,e} &= \int_{\kappa_e} N_e^T \bar{\boldsymbol{b}}_e \, \mathrm{d}\Omega + \int_{\partial\kappa_e} N_e^T \bar{\boldsymbol{t}}_e \, \mathrm{d}\Gamma
\end{aligned} \tag{1.10}$$

1.1.3 *Time integration of a discretized Finite Element problem*

The time integration of the solid problem is here carried out by a Generalized-α method [Chung and Hulbert, 1994]. In [Dettmer and Perić, 2003], the use of Finite Elements in time for the structure part of a fluid-structure interaction problem is shown to increase the computational cost of the overall problem for no noticeable advantages. The Generalized-α method is shown to be sufficiently robust and efficient for this range of application.

The time interval $[0,T]$ is discretized into a finite number of time-steps t_N such as $t_0 = 0$ and $t_{N_{\max}} = T$. The time step size is measured as $\Delta t = t_{N+1} - t_N$ and the time derivative are approximated with:

$$\begin{aligned}
\mathbf{u}_{N+1} &= \mathbf{u}_N + \Delta t\dot{\mathbf{u}}_N + \Delta t^2\left[\left(\frac{1}{2} - \beta\right)\ddot{\mathbf{u}}_N + \beta\ddot{\mathbf{u}}_{N+1}\right] \\
\dot{\mathbf{u}}_{N+1} &= \dot{\mathbf{u}}_N + \Delta t\left[(1-\gamma)\ddot{\mathbf{u}}_N + \gamma\ddot{\mathbf{u}}_{N+1}\right] \\
\mathbf{u}_{N+\alpha_f} &= (1-\alpha_f)\mathbf{u}_N + \alpha_f\mathbf{u}_{N+1} \\
\ddot{\mathbf{u}}_{N+\alpha_m} &= (1-\alpha_m)\ddot{\mathbf{u}}_N + \alpha_m\ddot{\mathbf{u}}_{N+1}
\end{aligned} \tag{1.11}$$

In the semi-discrete form of the solid equilibrium Eq. (1.9), the acceleration $\ddot{\mathbf{u}}$ and the displacement $\mathbf{u}$ are evaluated at $t_{N+\alpha_f}$ and $t_{N+\alpha_m}$ For the elastic linear

case, it is shown that there are optimum values for the parameters β, γ, α and α for a given spectral radius $\rho_\infty \in [0,1]$.

$$\beta = \frac{(1+\alpha_m-\alpha_f)^2}{4}, \gamma = \frac{1}{2}+\alpha_m-\alpha_f, \alpha_f = \frac{1}{1+\rho_\infty} \text{ and } \alpha_m = \frac{2-\rho_\infty}{1+\rho_\infty} \quad (1.12)$$

The spectral radius controls the numerical damping of the time integration scheme. The damping decreases with smaller values of ρ_∞ which is maximum for $\rho_\infty = 0$. For $\rho_\infty = 1$ the method is the classic trapezoidal rule. Other time integration schemes like α-HHT can be easily derived from this general formulation [Hilber et al., 1977].

1.2 Incompressible viscous flows solved by Finite Volume

1.2.1 *Strong form of the Navier-Stokes equations*

We consider a fluid domain Ω_f where an incompressible, viscous, isothermal and isotropic Newtonian flow is taking place. The Eulerian description of this flow in term of continuity (mass conservation) and momentum equilibrium gives us the following equations:

$$\begin{aligned} \partial_t \boldsymbol{v} + \nabla \cdot \boldsymbol{v} \otimes \boldsymbol{v} - \nabla \cdot 2\nu_f \mathbf{D}(\boldsymbol{v}) &= -\frac{1}{\rho}\nabla p \quad \text{in } \Omega_f \times [0,T] \\ \nabla \cdot \boldsymbol{v} &= 0 \quad \text{in } \Omega_f \times [0,T] \end{aligned} \quad (1.13)$$

where p denotes the kinematic pressure field, $\boldsymbol{v}$ the velocity. The constitutive law of the flow links the stress and the symmetric part of the velocity gradient $\mathbf{D}(\boldsymbol{v})$ through the kinematic viscosity ν_f which is the dynamic viscosity μ_f divided by the fluid density ρ_f.

Added to these equations, one also has to consider boundary equations at $\Gamma_f = \partial\Omega_f$. They can be of the Dirichlet or Neumann kind. One also has to set the initial conditions for the velocity and pressure fields in the whole domain Ω_f. Traditionally, the following physical boundary condition can be considered for the solving of a fluid problem:

No-slip boundary condition: can be chosen when viscosity is considered for the Navier-Stokes equations. For all surfaces of this kind, the fluid particles will be stuck, and therefore follow the motion if the boundary is moving:

$$\boldsymbol{v} = \overline{\boldsymbol{v}} \qquad \text{on } \partial\Omega_{f,D} \quad (1.14)$$

It typically leads to a so-called boundary layer where the flow is dominated by internal friction.

Slip-boundary condition: are traditionally chosen if no viscosity is considered (Euler equations), or for symmetric purposes of the fluid problem in the viscous case. In this case, only the normal flow is prescribed:

$$\boldsymbol{v} \cdot \boldsymbol{n} = \overline{\boldsymbol{v}} \cdot \boldsymbol{n} \qquad \text{on } \partial\Omega_{f,D} \quad (1.15)$$

Outflow boundary condition: are a particular challenge. The need to limit the computation domain to a finite size leads to consider "non-physical" boundary conditions for some parts of the Eulerian grid. For compressible flows, the well-known plane wave approximation can be used [Graff, 1975]. For the incompressible cases outflow boundary conditions are discussed in [Heywood et al., 1996]. In our case, the "do-nothing" is used. More precisely, a zero gradient Neumann boundary condition is applied for the fluid equation at the outflow:

$$\nabla \boldsymbol{v} \cdot \boldsymbol{n} = \boldsymbol{0} \qquad \text{on } \partial\Omega_{f,N} \tag{1.16}$$

At this stage, two remarks of importance, especially when one wants to consider the fsi domain of application, can be made on the pressure p. First, in Eq. (1.13), the pressure is divided by a constant density ρ. In most of the CFD solvers, the field computed as pressure is in fact p/ρ_f, and the only material property required by the computation is the kinematic viscosity ν_f. Thus, both the pressure induced-forces $p\boldsymbol{n}$ and the viscous forces $\nu\mathbf{D}(\boldsymbol{v})\boldsymbol{n}$ at the fluid-structure interaction boundary condition obtained from fluid computation have to be multiplied by the desired density ρ_f. Second, in the Navier-Stokes equation, only the gradient of the pressure ∇p is required. Therefore, pressure is determined when fixing the integration constant. To fix this value, either the mean value of the pressure, or its value at a special point, or at a boundary has to be imposed [Zienkiewicz and Taylor, 2001a]. The influence of this choice is discussed in my first numerical example for fluid-structure interaction in Section 4.1.

1.2.2 *Navier-Stockes equations discretization*

Three techniques are mainly used to solve these equations:

Finite Difference Method (FDM) starts from the conservation equations written in differential forms. The solution domain is covered by a grid, and at each point the differentials are approximated using the neighboring values. Such a technique can be used for every kind of mesh, but is most suitable for regular grids. Its main disadvantage is that conservation is not enforced unless special care is taken.

Finite Volume Method (FVM) can be formulated using integral forms of the conservation equations. Surface and volume integrals are approximated using suitable formulæ.

Finite Element Method (FEM) in which a predetermined base of functions is used to enforce the conservation equation in a weak sense.

For all these techniques, the continuous equations are transformed into a set of algebraic equations that can be solved numerically.

Finite Volume Methods is used here. Indeed, the goal of this Ph. D. work is first to emphasize the possibility of coupling different codes, with different methods (FEM for solid, FVM for fluid) and second to follow the trends of the computational scientific software market. And, as FVM, contrary to FEM,

leads to intrinsically conservative methods they are often preferred in commercial and non-commercial softwares (Phoenics, FLuent, FLow3D, Star-CD, Code_Saturne, OpenFOAM are FVM based, Adina is FEM based). Books of reference on the topic are [Patankar, 1980, Ferziger and Perić, 2002].

The FV formulation can be written directly using an integrated form of the conservation equation (1.13) [Jasak and Tuković, 2007]. Another possibility is to consider the restriction of the solution space of weak form problems. For uniformity with the solid method we chose to describe the FV strategy in this framework. The weak form of the Navier-Stokes equation can be written as follows [Glowinski et al., 2003]:

Find: $(\boldsymbol{v}, p) \in \mathcal{V} \times \mathcal{P}$, such that,
For all: $(\delta\boldsymbol{v}, \delta p) \in \mathcal{V}_0 \times \mathcal{P}_0$

$$\begin{aligned}\mathcal{G}_f(\boldsymbol{v}, p; \delta\boldsymbol{v}, \delta p) &:= \int_{\Omega_f} \partial_t \boldsymbol{v} \cdot \delta\boldsymbol{v} + \int_{\Omega_f} \nabla\boldsymbol{v} \otimes \boldsymbol{v} \cdot \delta\boldsymbol{v} - \int_{\Omega_f} \nabla \cdot \nu_f \mathbf{D}(\boldsymbol{v}) \cdot \delta\boldsymbol{v} \\ &\quad + \left(\int_{\Omega_f} \frac{1}{\rho_f} p \nabla \cdot \delta\boldsymbol{v} + \int_{\Omega_f} \frac{1}{\rho_f} \nabla \cdot \boldsymbol{v} \delta p \right) \\ &= 0\end{aligned}$$

The velocity field and the pressure are searched in $\mathcal{V}$ and $\mathcal{P}$ that denote the restrictions of a Hilbert space $\left[\mathcal{H}^1(\Omega_s)\right]^d$ and a Sobolev space $\left[\mathcal{L}^2(\partial\Omega_s, N)\right]^d$ with suitable boundary condition.

For this method, the whole volume Ω_f is divided into a set of of discrete elements, here called discrete volumes $(\kappa_{f,e})_{e=1,n_{el}}$ so that the whole domain is fulfilled ($\Omega_f = \cup_{e=1}^{n_{el}} \kappa_{f,e}$) without overlapping ($\cap_{e=1}^{n_{el}} \kappa_{f,e} = \emptyset$). For a Finite Element discretization, the solution space $\mathcal{V}$ and $\mathcal{P}$ are restricted to suitable spaces of piecewise polynomial functions over the set of discrete elements. For a Finite Volume discretization, the test functions are chosen in the space of characteristic discrete volume functions:

$$\mathcal{V}^h = \mathcal{V} \cap \left\{ \boldsymbol{v} \middle| \boldsymbol{v}|_\kappa \in \text{span}(\iota_\kappa), \forall \kappa \in \mathcal{T}^h(\Omega_f) \right\} \tag{1.17}$$

where ι_κ is the characteristic function of the element defined as:

$$\begin{aligned}\iota_\kappa : \ \Omega_f &\longrightarrow \mathbb{R} \\ \boldsymbol{x} &\longrightarrow \begin{cases} 1 & \text{if} \quad \boldsymbol{x} \in \kappa \\ 0 & \text{if} \quad \boldsymbol{x} \in \Omega/\kappa \end{cases}\end{aligned} \tag{1.18}$$

The same restriction holds for the pressure space $\mathcal{P}$ and the associated vector spaces $\mathcal{V}_0$ and $\mathcal{P}_0$. Therefore, the function are piecewise constant by elements, and the restriction of the weak formulation gives:

Find: $(\boldsymbol{v}^h, p^h) \in \mathcal{V}^h \times \mathcal{P}^h$, such that,
For all: $(\delta\boldsymbol{v}^h, \delta p^h) \in \mathcal{V}_0^h \times \mathcal{P}_0^h$

$$\mathcal{G}_f(\boldsymbol{v}^h, p^h; \delta\boldsymbol{v}^h, \delta p^h) = 0 \tag{1.19}$$

The divergence terms can be written in terms of flux at the boundary of volume controls using Gauss's theorem. Hence, the weak formulation restricted to this search space gives:

$$\begin{aligned}
&\mathcal{G}_f(\boldsymbol{v}^h, p^h; \delta\boldsymbol{v}^h, \delta p^h) \\
&= \sum_\kappa \left\{ \delta v_\kappa \left(\int_\kappa \partial_t \boldsymbol{v} \, \mathrm{d}\Omega - \oint_{\partial\kappa} \mathrm{d}\boldsymbol{\Gamma} \cdot \boldsymbol{v} \otimes \boldsymbol{v} + \oint_{\partial\kappa} \mathrm{d}\boldsymbol{\Gamma} \cdot 2\nu \mathbf{D}(\boldsymbol{v}) + \oint_{\partial\kappa} \frac{1}{\rho} p \, \mathrm{d}\boldsymbol{\Gamma} \right) \right\} \\
&+ \sum_\kappa \left\{ \delta p_\kappa \left(\oint_{\delta\Gamma_f} \mathrm{d}\boldsymbol{\Gamma} \cdot \boldsymbol{v} \right) \right\}
\end{aligned}$$

where $\mathrm{d}\boldsymbol{\Gamma}$ is the elementary surface vector. As one can notice, there is no continuity requirement for the solution (contrary to classical FE), and therefore the approached solutions $\boldsymbol{v}^h, p^h$ are not properly defined at the interface. Their flux can be computed without being imposed by the restriction of the solution space to the FV space. The whole difficulty is now to build an accurate representation of the fluxes at the boundaries from a piecewise constant field.

On each control volume, three levels of numerical approximations are applied to build the fluxes:

INTERPOLATION: to express variable values at the control volume surface in terms of nodal values (depending on where the variable is stored).

DIFFERENTIATION: to build convective and diffusive fluxes the value of the gradient of the quantity of interest – or at least its approximation – is required.

INTEGRATION: to approximate surface and volume integral using quadrature formulæ.

The details of building an accurate representation of the derived fields and especially of the convection terms (that require often special cares) will not be exposed here. Some schemes propose to stabilize the solution at the expense of accuracy (like the upwind scheme), others, such as the MUSCL (Monotone Upstream-centered Schemes for Conservation Laws) exhibit good stability properties for a high-order scheme [Piperno, 2000, Ferziger and Perić, 2002].

The d components of velocity v_i and pressure p are coupled through with a set of non-linear equations. Another way to see the problem is to look at it as a non-linear momentum equation under constraint. The pressure p is the Lagrangian multiplier and needs to be corrected to satisfy continuity on $\boldsymbol{v}$.

Written in a matrix forms, it gives the following semi-discrete problem for the discretized velocity $\mathbf{v}$ and pressure p field:

$$\begin{aligned}
\mathbf{M}_f \dot{\mathbf{v}} + \mathbf{N}_f(\mathbf{v})\mathbf{v} + \mathbf{K}_f \mathbf{v} + \mathbf{B}_f \mathsf{p} &= \mathbf{f} \\
\mathbf{B}_f^T \mathbf{v} &= \mathbf{0}
\end{aligned} \tag{1.20}$$

where $\mathbf{M}_f$ is a positive definite mass matrix, $\mathbf{N}_f$ is an unsymmetrical advection matrix, $\mathbf{K}_f$ describes the diffusion terms, and $\mathbf{B}_f$ stands for the gradient matrix, whereas $\mathbf{f}_f$ is the discretized load on the flow. This matrix form takes also into account the boundary conditions; special care has to be taken concerning the discretization of boundary conditions – and especially normal flux – when using the Finite Volume [Ghidaglia and Pascal, 2005].

A way to solve this problem is to consider a monolithic application. An other way is to consider a split between the momentum and the continuity equations, and to solve it thanks to an operator split-like procedure often termed as the *segregated approach* [Patankar, 1980]. This approach is favored for its computational efficiency compared to the monolithic one. Indeed, even with a simple fixed point iteration strategy its cost is less important than that of the monolithic approach for large size problems [Ferziger and Perić, 2002]. By default, the Navier-Stokes equation in our `OpenFOAM` based component is solved with the second one, and is detailed in the next Section.

1.2.3 *Pressure-Implicit with Splitting of Operators (PISO) algorithm*

The Finite Volume Method has good properties in term of conservation, but even if high order derivatives can be built using suitable approximations, a precise computation often requires a large number of d-o-f. For that reason, building the Jacobian of the whole coupled $(\mathbf{v}, \mathsf{p})$ problem is rarely done. Traditional algorithms rely on the splitting of operators associated with the velocity and pressure parts, solved in an explicit, implicit or half-implicit way. Thus, the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) and the PISO (Pressure Implicit with Splitting of Operators) algorithms are traditionally used in CFD. The first one heavily relies on the use of suitable relaxation parameters [Ferziger and Perić, 2002] in order to reach the same order of efficiency as the latest. In [Barton, 1998], a comparison between the two algorithms show the overall better performances of PISO-like algorithms over SIMPLE ones.

In this work, PISO-like algorithms are used to solve our CFD problem. The semi-discrete form of the Navier-Stokes equation is discretized in time using implicit or explicit integration schemes, such as Euler explicit and implicit, or a second order Crank-Nicholson scheme. The discretized momentum Eq. (1.20) is split in the following way when an implicit integration scheme is used:

$$\mathcal{A}_f(\mathbf{v}_{N+1})\mathbf{v}_{N+1} - \mathcal{H}_f(\mathbf{v}_N, \mathbf{v}_{N+1}) = -\mathbf{B}_f \mathsf{p}_{N+1} \tag{1.21}$$

where $\mathcal{A}_f$ stands for time derivative terms in a cell (and is therefore diagonal) and $\mathcal{H}_f$ takes into account all neighboring velocity and source terms in elements. The incompressibility condition can be re-written in a discrete form using the previous split as:

$$\mathbf{B}_f^T \frac{1}{\mathcal{A}_f(\mathbf{v}_{N+1})} \mathbf{B}_f \mathsf{p}_{N+1} - \mathbf{B}_f^T \frac{1}{\mathcal{A}_f(\mathbf{v}_{N+1})} \mathcal{H}_f(\mathbf{v}_N, \mathbf{v}_{N+1}) = \mathbf{0} \tag{1.22}$$

For the PISO algorithm, the coupling between the incompressibility condition and the momentum equilibrium parts of the Navier-Stokes equation is assured in an iterative way as detailed in **Algorithm 1**.

The PISO algorithm is not fully implicit, as corrective term of the velocity is introduced explicitly. Hence, in the correction step, it is supposed that the influence of the transported term is negligible compared to the pressure gradient correction terms. Therefore, even with an implicit time integration scheme, the stability of the PISO algorithm remains conditional, and when the Courant Number becomes too large (it means that the transport due to the flow over a cell is not well captured) the PISO algorithm fails to converge.

Algorithm 1 Pressure Implicit Splitting of Operators algorithm for fluid

1: Given: initial velocity $\mathbf{v}_0$ and pressure p_0.
2: **while** $T < T_{\max}$ **do**
3: Momentum predictor:

$$\mathcal{A}_f(\mathbf{v}_{N+1}^{(0)})\mathbf{v}_{N+1}^{(0)} - \mathcal{H}_f(\mathbf{v}_N, \mathbf{v}_{N+1}^{(0)}) = -\mathbf{B}_f \mathsf{p}_N$$

4: **for** $(k) = 0$ to $(k) < (k_{\max})$ **do**
5: Pressure correction:

$$\mathbf{B}_f^T \cdot \frac{1}{\mathcal{A}_f(\mathbf{v}_{N+1}^{(k)})} \mathbf{B}_f^T \mathsf{p}_{N+1}^{(k)} = \mathbf{B}_f^T \cdot \frac{\mathcal{H}_f(\mathbf{v}_N, \mathbf{v}_{N+1}^{(k)})}{\mathcal{A}_f(\mathbf{v}_{N+1}^{(k)})}$$

6: Explicit velocity correction

$$\mathbf{v}_{N+1}^{(k+1)} = \mathbf{v}_{N+1}^{(k)} - \frac{1}{\mathcal{A}(\mathbf{v}_{N+1}^{(k)})} \mathbf{B}_f \mathsf{p}_{N+1}^{(k)}$$

7: **end for**
8: **end while**

REMARK: in the algorithm given, the non-orthogonal correctors applied to build more accurate flux terms when the mesh is not orthogonal grid are not detailed. For further details, see [Jasak, 1996, Ferziger and Perić, 2002].

1.2.4 *Validating the CFD strategy*

Computational Fluid Dynamics aims at solving equations whose solution are not reachable by analytical ways. Furthermore, comparing the results of numerical simulations with experiments is not an easy task, as a result the equivalence between the problems studied cannot be perfect. To know if a solver gives good results, a first way can be to use an error estimate [Jasak, 1996]. A good way remains however to perform simulations defined in benchmark and to compare their results with the ones obtained by other teams.

In [Schäfer and Turek, 1996], benchmarks for steady/unsteady 2D/3D flow simulations are proposed. The results from teams working independently from one another are compared, and even for the presumed simple case proposed, it was shown that solutions are not always in a close range. In this section, the results obtained by a Finite Volume discretization of the Navier-Stokes equation, and solved thanks to PISO algorithm are compared to the reference solution from these benchmarks.

1.2.5 *Two-dimensional numerical experiments*

1.2.5.1 *Problem description*

The benchmark proposes to study the laminar flow around a cylinder. The dimension of the problem as well as the boundary condition are defined in Fig. 1.3.

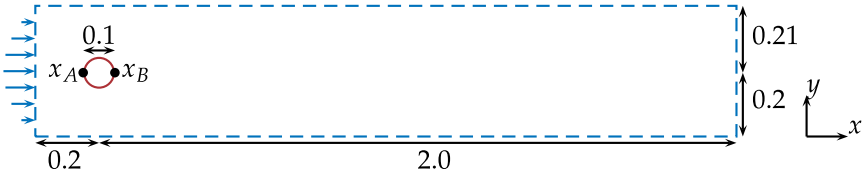


Figure 1.3: Fluid benchmark geometry and boundary conditions

For these bluff body problems, it is traditional to compute the fluid forces acting on the obstacle coming from the deviatoric (pressure) part and its complement (the viscous forces):

$$\boldsymbol{F}_f = \rho_f \left(-p\,\boldsymbol{n} + \nu_f \mathbf{D}(\boldsymbol{v})\,\boldsymbol{n} \right) \tag{1.23}$$

with $\mathbf{D}(\boldsymbol{v}) = \frac{1}{2}\left(\nabla \boldsymbol{v} + \nabla \boldsymbol{v}^T\right)$ the symmetric part of the velocity gradient tensor. These forces can be projected along the main flow direction to obtain the drag force:

$$F_{\mathrm{D}} = \int_{\Gamma} \boldsymbol{F}_f \cdot \boldsymbol{e}_x \, \mathrm{d}\Gamma \tag{1.24}$$

and perpendiculary to get the lift force:

$$F_{\mathrm{L}} = \int_{\Gamma} \boldsymbol{F}_f \cdot \boldsymbol{e}_y \, \mathrm{d}\Gamma \tag{1.25}$$

In fluid mechanics, it is traditional to work with dimensionless values. The associated two non-dimensional values are the lift coefficient:

$$\mathcal{C}_{\mathrm{L}} = \frac{2F_{\mathrm{L}}}{\rho_f \|\overline{\boldsymbol{v}}\|^2 D} \tag{1.26}$$

and the drag coefficient:

$$\mathcal{C}_{\mathrm{D}} = \frac{2F_{\mathrm{D}}}{\rho_f \|\overline{\boldsymbol{v}}\|^2 D} \tag{1.27}$$

where D is a characteristic dimension for the flow (here the diameter of the cylinder obstacle) and $\overline{\boldsymbol{v}}$ the imposed velocity at the flow input.

The kind of flow is also characterized by its Reynolds number, that gives the ratio between convective and diffusive part in the Navier-Stokes momentum conservation equation:

$$\mathcal{R}e = \frac{D\|\overline{\boldsymbol{v}}\|}{\nu_f} \tag{1.28}$$

For a certain range of Reynolds numbers, the flow exhibits vortex shedding. The vortex shedding frequency f, that can be measured with the lift and drag

coefficient oscillation around their mean values is also traditionally adimensioned with the Strouhal number:

$$\mathcal{S}t = \frac{Df}{\overline{v}_x} \tag{1.29}$$

The Finite Volume strategy used herein is second order in space (flux construction) and a first order in time (Euler implicit time integrator). The flow solver used is far from being optimum in terms of computational time, as no special care is take for the pseudo time step-size, for the inverse algorithm (here the default iterative algorithm like a Pre-conditioned Conjugate Gradient is used) and for the parallelization. So, the CPU time T^{CPU} is only given for indication, but must not be considered as a good overall performance indicator of the fluid solver.

1.2.5.2 *Test case 2D-1 (steady)*

The inflow condition is:

$$\boldsymbol{v}(x=0,y,t) = \left(U_{\max}\frac{4y(H-y)}{H^2}, 0\right) \tag{1.30}$$

with $U_{\max} = 0.3m.s^{-1}$ leading to a Reynold number $\mathcal{R}e = 20$. Such a Reynold number characterizes a laminar and steady flow. For the coarse mesh cases, less than 300 pseudo time-steps were necessary to converge to the solution. For finer mesh cases, it becomes necessary to compute at initial time a potential solution. The comparison with the benchmark is given for adimensional lift and drag coefficients and for the pressure difference between the front and back nodes of the obstacle.

Elements	$\mathcal{C}_{\text{D}}$	$\mathcal{C}_{\text{L}}$	Δp (in Pa)	T^{CPU} (in s)
528	5.589	0.0116	0.1109	1.37
2112	5.603	0.0132	0.1140	4.13
8448	5.597	0.0111	0.1169	16.59
33792	5.588	0.0107	0.1172	384.83
Ref.	5.560 $\pm$0.010	0.0107 $\pm$0.0003	0.1174 $\pm$0.0002	

Table 1.1: Fluid benchmark test 2D-1: `OpenFOAM` results compare to reference solution

In Tab. 1.1 some of the quantities obtained from the flow computation such as lift coefficient $\mathcal{C}_{\text{L}}$, drag coefficient $\mathcal{C}_{\text{D}}$ and pressure difference noted Δp between the nodes $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ are given. The results obtained and summarized on Tab. 1.1 converge to accordance with the benchmark reference values given in [Schäfer and Turek, 1996].

1.2.5.3 *Test case 2D-2 (unsteady)*

For this second test case, the input velocity is increased in order to obtain an unsteady flow:

$$\boldsymbol{v}(x=0,y,t) = \left(U_{\max}\frac{4y(H-y)}{H^2}, 0\right) \tag{1.31}$$

with $U_{\max} = 1.5 m.s^{-1}$ leading to a Reynold number $\mathcal{R}e = 100$. Such flows exhibit vortex shedding: drag coefficient $\mathcal{C}_{\mathrm{D}}$, lift coefficient $\mathcal{C}_{\mathrm{L}}$ and pressure difference Δp become periodic functions of time t.

Elements	$\mathcal{C}_{\mathrm{Dmax}}$	$\mathcal{C}_{\mathrm{Lmax}}$	$\mathcal{S}t$	Δp (in Pa)	T^{CPU} (in s)
528	3.25431	1.1216	0.2341	2.3667	14.65
2112	3.17842	1.0379	0.2830		129.15
8448	3.21425	0.9819	0.2929		2833.90
37792	3.23270	1.0152	0.3052	2.5156	35472.83
Ref.	3.2300 $\pm$0.0100	1.0000 $\pm$0.0100	0.3000 $\pm$0.0050	2.4800 $\pm$0.0200	

Table 1.2: Fluid benchmark test 2D-2: `OpenFOAM` results compared to reference solution

The quantity used for comparison with the Benchmark are: maximum lift coefficient $\mathcal{C}_{\mathrm{Lmax}}$, maximum drag coefficient $\mathcal{C}_{\mathrm{Dmax}}$, Strouhal number $\mathcal{S}t$ and pressure difference noted Δp between the nodes $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$ when the lift coefficient is minimum. The results obtained are summarized in Tab. 1.2 correspond to the results of reference.

1.2.5.4 *Test case 2D-3 (unsteady)*

In the previous case, setting suitable initial condition can be a problem. Thus, in this case, the flow is initially at rest and the input velocity is gradually increased as follows:

$$\boldsymbol{v}(x = 0, y, t) = \left(U_{\max} \frac{4y(H-y)}{H^2} \sin \frac{\pi t}{8}, 0 \right) \tag{1.32}$$

with $U_{\max} = 1.5 m.s^{-1}$ and for $t \in [0,\ 8s]$. This leads to a time dependant Reynolds number from 0 to 100.

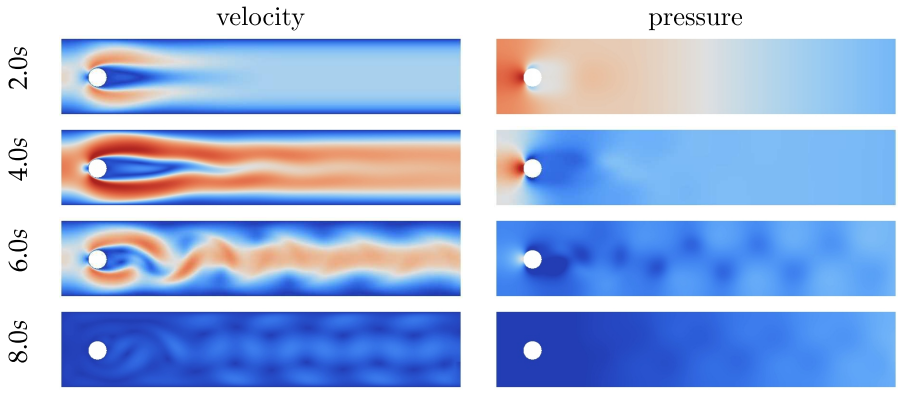


Figure 1.4: Test 2D-3: Velocity (magnitude) and pressure fields

In Fig. 1.4, the velocity and pressure field for different times are represented. The problem evolves from a quasi-steady state (*e.g.* $t = 2.0s$) to an unsteady

problem with vortex shedding (*e.g.* $t = 4.0s$). The quantity needed for comparison with the Benchmark are : maximum lift coefficient $\mathcal{C}_{\mathrm{Lmax}}$, maximum drag coefficient $\mathcal{C}_{\mathrm{Dmax}}$, and pressure difference at end time $\Delta p(t = 8s)$.

Elements	$\mathcal{C}_{\mathrm{Dmax}}$	$\mathcal{C}_{\mathrm{Lmax}}$	Δp (in Pa)	T^{CPU} (in s)
528	2.9571	0.3143	-0.1018	13.8
2112	2.8964	0.5099	-0.1045	101.4
8448	2.9354	0.4753	-0.1023	1915.07
37792	2.9469	0.4947	-0.1106	32250.88
Ref.	2.9500	0.4800	-0.1100	
	± 0.0200	± 0.0100	± 0.0050	

Table 1.3: Fluid benchmark test 2D-3: `OpenFOAM` results compared to reference solution

In Tab. 1.3 the test results are displayed. Computation are carried out with OpenFOAM-1.5 icoFoam solver – for Newtonian flows with a low Reynolds number – on only one processor (Intel Xeon X5365 3.00GHz). The time integration scheme was Crank-Nicholson with $\alpha = 0.7$ and the time step is decreased in order to respect the CFL condition required by the PISO algorithm, and varies from $0.01s$ for the coarsest mesh to $0.0005s$ for the finest one.

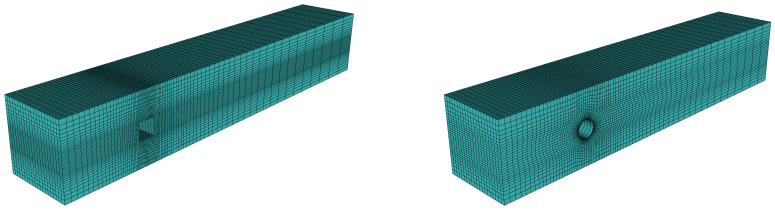
1.2.6 *Three-dimensional numerical experiments*

As one of our goals is to solve fluid-structure interaction problems in three dimensions, the validation of the CFD approach in three dimensions is also carried out for equivalent test cases. We present herein the results for the steady state flow around a square basis (3D-1Q) and circular basis (3D-1Z) -shaped cylinder. The dimensions on the $(\boldsymbol{e}_x, \boldsymbol{e}_y)$ plane are the same as the one defined in Fig. 1.3. The third dimension is chosen so that the inflow and outflow are defined on square plane. One can notice that even in [Schäfer and Turek, 1996], the results from the different teams for the three dimensional tests were not in agreement for all of them. Some three dimensional meshes used for computation are represented in Fig. 1.5.

The time integration scheme used was backward Euler with a pseudo time-step of 1s and with initial condition at rest. The steady state is quickly reached, and in Fig. 1.6 some characteristic streamlines are represented. The computations is carried out on four meshes, and the results are given in Tab. 1.4.

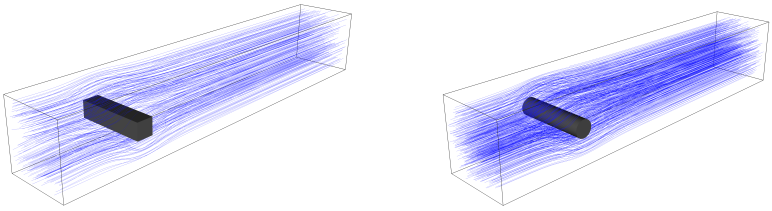
The results from Tab. 1.4 are not so much in agreement as the ones from the reference. For the 3D cases, contrary to the 2D cases, the results from the different teams benchmarked in [Schäfer and Turek, 1996] is more marked than the one given in for 2D computations. The difference from the referring solutions is most marked for the square cylinder, probably due to more sensitiveness of the flow around this kind of shape.

Now that the Finite Volume Strategy used for the fluid is presented and benchmarked, the range of application is extended in the next Section to flow in moving domains.



(a) 3D-1Q problem (b) 3D-1Z problem

Figure 1.5: Two meshes for the 3D-1Q and 3D-1Z test. Medium sized mesh with around 50×10^3 cells.



(a) 3D-1Q problem (b) 3D-1Z problem

Figure 1.6: Streamlines for the 3D-1Q and 3D-1Z test.

1.3 Flow in a moving shape domain

Fluid-Structure interaction problems lead to unsteady moving domains for the fluid part, as the fluid-structure interface follows the deformations of the structure. Traditional Computational Fluid Dynamic programs solve the fluid equations on a fixed (Eulerian) grid.

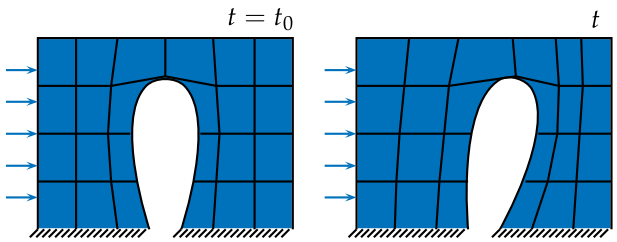


Figure 1.7: Fluid in a moving domain problem

A classic approach to overcome this difficulty is to consider the so-called

Test 3D-1Q

Elements	$\mathcal{C}_D$	$\mathcal{C}_L$	Δp (in Pa)	T^{CPU} (in s)
7760	6.09	0.0784	0.1121	2.88
26190	6.76	0.0734	0.1261	29.54
62080	6.57	0.0780	0.1204	102.09
209520	7.29	0.1653	0.1385	1244.89
Ref.	7.60 $\pm$0.10	0.0700 $\pm$0.0100	0.1760 $\pm$0.0040	

Test 3D-1Z

Elements	$\mathcal{C}_D$	$\mathcal{C}_L$	Δp (in Pa)	T^{CPU} (in s)
6120	5.72	0.01143	0.1518	4.75
20655	5.95	0.01109	0.1587	21.43
48960	6.06	0.01046	0.1653	70.17
165240	6.14	0.01025	0.1683	925.19
Ref.	6.15 $\pm$0.10	0.0090 $\pm$0.0010	0.1700 $\pm$0.0050	

Table 1.4: Fluid benchmark test 3D-1Q and 3D-1Z: `OpenFOAM` results compared to reference solution

Arbitrary Lagrangian Eulerian (ALE) method where the grid is moved inside the fluid domain, following the movement of the boundary (see Fig. 1.7).

However, this yields new difficulties: knowing the new shape of the boundary, how can one keep the quality and the validity of the inner mesh? A traditional and widely used method is to consider the mesh as a pseudo-structural system. Mesh points are linked with kind of spring. Another solution is to describe the mesh by analogy as a solid loaded at its boundary.

This pseudo-structural system is known to be unsuccessful with too large motion cases. This is why one often has to consider non-linear pseudo structural systems and sometimes needs to couple them with re-meshing algorithms. Even using additional concepts to reduce the computational cost, this procedure remains expensive and does not solve all the problems. Recent methods relying on the use of linear Laplace equations, which can give bounded solutions, present reliable mesh motion for a reasonable cost [Jasak and Tuković, 2007].

The ALE strategy leads in fact to a three field coupling (the fluid, the structure and the mesh motion). However, modern CFD codes often propose the implementation of ALE. Furthermore, a fluid-structure interaction problem does not only become a two-field (fluid and solid) but also a three-field coupling problem (fluid, solid and fluid domain). Besides, the global solution must not depend on the mesh motion, and this is naturally verified when the quality of the mesh is preserved.

The limitation of some mesh motion algorithms are studied and highlighted in [Kassiotis, 2008][1]. From this work, one can recall that:

LAPLACE SMOOTHING OPERATOR with a non constant diffusion coefficient can handle large deformation of the mesh while the rotations remains small.

[1] See also the reference thread in http://www.cfd-online.com/Forums/openfoam-solving/57916-moving-mesh-problem-openfoam-141-a.html `OpenFOAM` forum

PSEUDO-SOLID FORMULATION is of interest when the motion of the mesh is mainly governed by rotation.

1.3.1 *Arbitrary Lagrangian-Eulerian strategy*

1.3.1.1 *Navier-Stokes equation in an Arbitrary Lagrangian-Eulerian framework*

For a given motion of the fluid domain Ω_f described by its point displacement $\boldsymbol{u}_m$, the ALE formulation of the Navier-Stokes equation can be written as:

$$\begin{aligned} \partial_t \boldsymbol{v} + (\boldsymbol{v} - \dot{\boldsymbol{u}}_m)\,\nabla\cdot\boldsymbol{v} - \nabla\cdot 2\nu_f \mathbf{D}(\boldsymbol{v}) &= -\frac{1}{\rho}\nabla p \quad \text{in } \Omega_f \times [0,T] \\ \nabla\cdot\boldsymbol{v} &= 0 \quad \text{in } \Omega_f \times [0,T] \end{aligned} \tag{1.33}$$

where $\boldsymbol{v} - \dot{\boldsymbol{u}}_m$ is the convective term. The Eulerian or Lagrangian expression of the total time derivative of $\boldsymbol{v}$ can be found setting $\dot{\boldsymbol{u}}_m = \boldsymbol{0}$ or $\dot{\boldsymbol{u}}_m = \boldsymbol{v}$. In the fluid-structure interaction framework, the motion of the interface near the structure imposes a convective governed flow, and its description is Lagrangian. Away from it, there is no reason to make the domain move, and the description is still Eulerian.

There remains the question of building a suitable map for the domain motion given its interface displacement. The fluid displacement $\boldsymbol{u}_m$ is arbitrary inside the domain Ω_f, but has to fulfill the condition:

$$\boldsymbol{u}_m = \overline{\boldsymbol{u}}_m \qquad \text{on } \partial\Omega_f \times [0,T] \tag{1.34}$$

Thus, the fluid displacement is an arbitrary extension of $\overline{\boldsymbol{u}}_m|_{\partial\Omega_f}$ inside the fluid domain Ω_f:

$$\boldsymbol{u}_m = \mathrm{Ext}\left(\overline{\boldsymbol{u}}_m\Big|_{\partial\Omega_f}\right) \tag{1.35}$$

The weak form of the ALE formulation can be written as follows:

Find: $(\boldsymbol{v}, p, \boldsymbol{u}_m) \in \mathcal{V}\times\mathcal{P}\times\mathcal{U}$, such that,
For all: $(\delta\boldsymbol{v}, \delta p, \delta\boldsymbol{u}) \in \mathcal{V}_0\times\mathcal{P}_0\times\mathcal{U}_0$

$$\begin{aligned} &\mathcal{G}_f^{\mathrm{ALE}}(\boldsymbol{v}, p, \boldsymbol{u}_m; \delta\boldsymbol{v}, \delta p, \delta\boldsymbol{u}) := \\ &\int_{\Omega_f}\left(\boldsymbol{u}_m - \mathrm{Ext}\left(\overline{\boldsymbol{u}}_m\Big|_{\partial\Omega_f}\right)\right)\cdot\delta\boldsymbol{u} \\ &\int_{\Omega_f}\partial_t\boldsymbol{v}\cdot\delta\boldsymbol{v} + \int_{\Omega_f}\nabla(\boldsymbol{v}-\dot{\boldsymbol{u}}_m)\otimes\boldsymbol{v}\cdot\delta\boldsymbol{v} - \int_{\Omega_f}\nabla\cdot\nu_f\mathbf{D}(\boldsymbol{v})\cdot\delta\boldsymbol{v} + \\ &\int_{\Omega_f}\frac{1}{\rho_f}p\nabla\cdot\delta\boldsymbol{v} + \int_{\Omega_f}\frac{1}{\rho_f}\nabla\cdot\boldsymbol{v}\delta p \\ &\quad = 0 \end{aligned}$$

Written in a matrix forms, it gives the following semi-discrete problem for

the discretized velocity $\mathbf{v}$ and pressure p field:

$$\begin{aligned} \mathbf{K}_m \mathbf{u}_m &= \mathbf{f}_m \\ \mathbf{M}_f \dot{\mathbf{v}} + \mathbf{N}_f(\mathbf{v} - \dot{\mathbf{u}}_m)\mathbf{v} + \mathbf{K}_f \mathbf{v} + \mathbf{B}_f \mathsf{p} &= \mathbf{f}_f \\ \mathbf{B}_f^T \mathbf{v} &= \mathbf{0} \end{aligned} \tag{1.36}$$

where $\mathbf{K}_m$ is a stiffness matrix for the mesh deformation, $\mathbf{M}_f$ is a positive definite mass matrix, $\mathbf{N}_f$ is an unsymmetrical advection matrix, $\mathbf{K}_f$ describes the diffusion terms, and $\mathbf{B}_f$ stands for the gradient matrix, whereas $\mathbf{f}_f$ is the discretized load on the flow. This matrix form takes also into account the boundary conditions; special care has to be taken concerning the discretization of boundary conditions – and especially normal flux – when using the Finite Volume [Ghidaglia and Pascal, 2005].

1.3.1.2 *Mesh motion equations*

Let us consider a domain $\mathcal{T}^h\left(\Omega_{f,t_N}\right)$ which represent the mesh configuration at a time t_N. This domain is moving according to the imposed condition at the boundaries like imposed displacement or velocity. The issue is to build a new valid mesh $\mathcal{T}^h\left(\Omega_{f,t_{N+1}}\right)$ at $t_{N+1} = t_N + \Delta t$ knowing the initial valid mesh $\mathcal{T}^h\left(\Omega_{f,t_N}\right)$ and imposed boundary conditions.

It can be proved that the mesh motion problem is formally equivalent to a solid body under large deformations. The cost to solve this non-linear equation is tremendous and this strategy cannot be applied within an efficient 3D CFD code. A traditional way to overcome this difficulty is to consider simplified solid equations for the mesh motion:

The spring analogy aims at linking each point of the mesh by fictitious spring. In [Jasak and Tuković, 2007], authors show this yields failure modes. This failure modes can be avoided introducing non-linear and torsional springs. However, the cost induced by the improvement of this imperfect system can be deemed as too huge.

pseudo-solid equations can be considered as a simplification of the 3D non-linear problem to the linear one; the analogy is a mechanical problem with small deformations.

Laplacian smoothing operator

A totally different strategy is to consider the Laplace smoothing equation:

$$\nabla \cdot (\gamma \nabla \boldsymbol{u}) = 0 \qquad \text{in} \quad \Omega_f \tag{1.37}$$

Where $\boldsymbol{u}$ is the mesh displacement so that the new points are defined with:

$$\boldsymbol{x}_{\mathrm{N+1}} = \boldsymbol{x}_{\mathrm{N}} + \Delta t \; \boldsymbol{u} \tag{1.38}$$

The Laplace smoothing equation does not allow to take into account coupling the motion vector components due to rotation. These coupled motion are handled by the *pseudo-solid* solver.

1.3.1.3 *PISO algorithm for Navier-Stokes equation discretized by Finite Volume in a moving domain*

The coupling between the mesh motion problem and the Navier-Stokes momentum equation is weak, in the sense that no variable like velocity $\boldsymbol{v}$ or pressure p influences the fluid domain deformation under imposed boundary displacements. The coupling between the mesh motion problem and the fluid momentum equation can therefore be ensured explicitly. In the PISO **Algorithm 1**, the only modification is to compute the mesh motion for the current time step before solving the semi-implicit coupled velocity-pressure system.

Algorithm 2 PISO algorithm for ALE formulated flows

Given: initial velocity v_0 and pressure p_0.
while $T < T_{\max}$ **do**
  Solve mesh motion:

$$\mathcal{R}_m(\boldsymbol{u}_{m,N+1}, \overline{\boldsymbol{u}}_{m,N+1}) = 0$$

  Momentum predictor:

$$\mathcal{A}_f(\boldsymbol{u}_m, \mathsf{v}_{N+1}^{(0)})\mathsf{v}_{N+1}^{(0)} - \mathcal{H}_f(\mathsf{v}_N, \boldsymbol{u}_m, \mathsf{v}_{N+1}^{(0)}) = -\mathbf{B}_f \mathsf{p}_N$$

  for $(k) = 0$ to $(k) < (k_{\max})$ **do**
    Pressure correction:

$$\mathbf{B}_f^T \cdot \frac{1}{\mathcal{A}_f(\boldsymbol{u}_m, \mathsf{v}_{N+1}^{(k)})} \mathbf{B}_f^T \mathsf{p}_{N+1}^{(k)} = \mathbf{B}_f^T \cdot \frac{\mathcal{H}_f(\boldsymbol{u}_m, \mathsf{v}_N, \mathsf{v}_{N+1}^{(k)})}{\mathcal{A}_f(\boldsymbol{u}_m, \mathsf{v}_{N+1}^{(k)})}$$

    Explicit velocity correction

$$\mathsf{v}_{N+1}^{(k+1)} = \mathsf{v}_{N+1}^{(k)} - \frac{1}{\mathcal{A}_f(\boldsymbol{u}_m, \mathsf{v}_{N+1}^{(k)})} \mathbf{B}_f \mathsf{p}_{N+1}^{(k)}$$

  end for
end while

The only remaining question is the choice of velocity in the time step $\Delta t = t_{N+1} - t_N$. As the mesh motion is arbitrary and does not rely on any physical phenomenon, it is *a priori* possible to take any velocity evolution on the window $[T_N, T_{N+1}]$ so that the initial mesh deformation is equal to $\boldsymbol{u}_{m,N}$ and the final mesh deformation is equal to $\boldsymbol{u}_{m,N+1}$.

The Geometric Conservation Law demands a numerical scheme to reproduce exactly and independently from the mesh motion a constant solution. This condition can be found in the literature for ALE formulation discretized either by the Finite Volume [Demirdžić and Perić, 1988] or the Stabilized Finite Element methods [Förster et al., 2006]. It is proved [Farhat et al., 2001] that the velocity

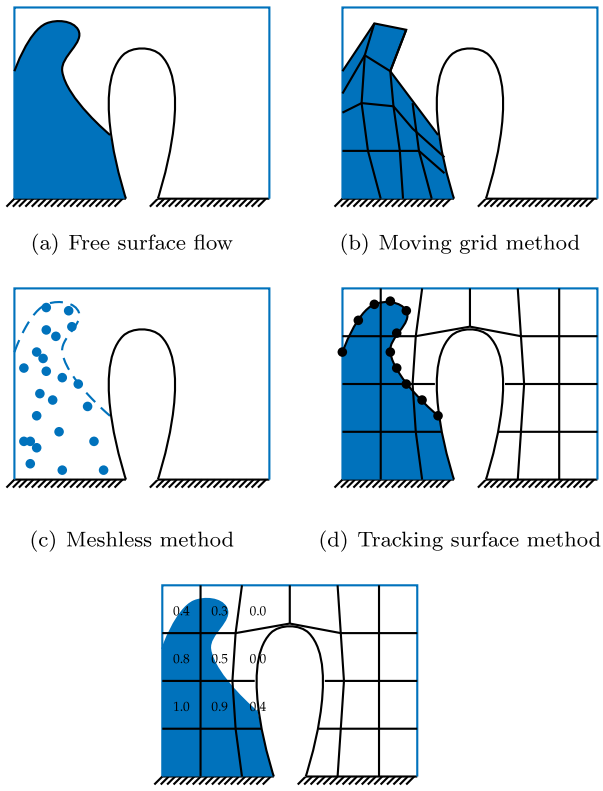
of the dynamic mesh needs to be computed for all first– and second–order time accurate methods like implicit Euler or Crank-Nicholson so that:

$$\dot{u}_m = \frac{\boldsymbol{u}_{m,N+1} - \boldsymbol{u}_{m,N}}{\Delta t} \tag{1.39}$$

1.3.2 *Free-surface flows*

1.3.2.1 *Strategies for free-surface flows*

To model free surface flows, a traditional approach is to consider the Nonlinear Shallow Water Equations first introduced by Saint-Venant in 1837. Extensive literature on how to solve those equations analytically or numerically can be found in the applied mathematics community [Stoker, 1992, Sobey, 1998]. If this approach gives good results for the propagation phase of the wave, recent results in [Dutykh and Mitsotakis, 2009] confirm that this model is unable to represent accurately the impact process of the wave, as well as its sloshing process (see Fig. 1.8(a)).



(a) Free surface flow (b) Moving grid method

(c) Meshless method (d) Tracking surface method

(e) Tracking volume method (VOF)

Figure 1.8: Techniques to solve a free-surface flow

For this reason, another appropriate model has to be chosen near a structure

hit by water. To represent the complex flow an idea could be to consider a moving fluid domain where the shape of the wave is described by the motion of the domain. However, the domain motion is so complicated that this strategy is not reachable with the traditional ALE formulation given above.

The following approaches are however used:

LAGRANGIAN APPROACHES: (see Fig. 1.8(a)) are used to represent the free surface fluid flows. Such approaches often require re-meshing, and therefore efficient mesh generators in order to compute the new topology at each iteration. The PFEM method [Idelsohn et al., 2004] is for instance used for this kind of problems.

SURFACE TRACKING METHODS (ALE): (see Fig. 1.8(d)) have the advantage that the representation of the fluid domain and its free surface is independent from the representation of the flow field. Hence, the resolution of the surface and that of the flow field may be chosen independently one from another. Of course, the level of details should be comparable in order to resolve the fluid motion properly. However, this freedom is helpful, for example, in order to improve the accuracy of the evaluation of the surface tension force.

MESHLESS METHODS (see Fig. 1.8(b)) like the SPH have the advantage not to require the re-meshing step of the previous strategies [Fries, 2005]. They are however still quite expensive, and limited for the time being to local computations.

Neither the first nor the second approaches are used herein. A Eulerian or Arbitrary Lagrangian-Eulerian volume method formulation of a biphasic flow modeling both water and surrounding air is used in the following [Ubbink, 1997, Ghidaglia et al., 2001, Rusche, 2002]. In Volume of Fluid (V.O.F.) methods, an indicator function (volume fraction, level set or phase-field) is used to represent the interface (see Fig. 1.8(e)); the issue is how to convect the interface without diffusing, dispersing or wrinkling it. This is particularly troublesome when the volume fraction is chosen as an indicator function because the convection scheme has to guarantee that the volume fraction stays bound, *i.e.* remains within its physical bounds of 0 and 1. This problem has been addressed by a number of authors and two distinct methodologies to convect the volume fraction have emerged [Rusche, 2002]: Volume of Fluid (V.O.F.) methods use convection schemes which reconstruct the interface from the volume fraction distribution before advecting it. On the other hand, interface-capturing techniques use high-order convection schemes.

1.3.2.2 *A biphasic model for free surface flows*

For complex flows (with jets, cavitation and aeration in the sloshing wave) it is natural to consider the Navier-Stokes equations for two immiscible and incompressible flows (water and air for instance) occupying transient domains $\Omega_i(t)$ so that the whole domain considered $\Omega(t) = \Omega_1(t) \cup \Omega_2(t)$. The interface between the both domains Ω_1 and Ω_2 is denoted Γ

In each space-time domain $\Omega_i \times [0,T]$, the Navier-Stokes equations apply:

$$\begin{aligned} \nabla \cdot \boldsymbol{v} &= 0 \\ \rho_i \partial_t \boldsymbol{v} + \nabla \cdot (\boldsymbol{v} \otimes \boldsymbol{v} + p\mathbf{I} - 2\mu_i \mathbf{D}(\boldsymbol{v})) &= \sigma\kappa\delta_\Gamma \boldsymbol{n} + \rho_i \boldsymbol{g} \end{aligned} \tag{1.40}$$

where p denotes the kinematic pressure field, $\boldsymbol{v}$ the velocity. Fluid material properties are the dynamic viscosities μ_i and the densities ρ_f. We also introduce $\boldsymbol{g}$ that depicts the gravity field, σ the surface tension, κ the curvature of the free-surface and δ_Γ the mass distribution concentrated at the surface (equivalent to a Dirac distribution).

REMARK: The incompressible hypothesis can be discussed, especially for highly aerated waves. For more details on compressible multiphasic simulations, the reader is invited to follow the Ph. D. work of [Dutykh, 2008] or [Dias et al., 2009].

For the Finite Volume approach, both fluid domains are computed with the same Navier-Stokes models. It is advantageous to introduce the the characteristic function ι:

$$\iota(\boldsymbol{x},t) = \begin{cases} 1, & \text{for} \quad \boldsymbol{x} \in \Omega_1(t) \\ 0, & \text{for} \quad \boldsymbol{x} \in \Omega_2(t) \end{cases} \tag{1.41}$$

Thus, the characteristic function ι and the mass distribution δ_Γ are linked by the relation:

$$\nabla \iota = \delta_\Gamma \boldsymbol{n} \tag{1.42}$$

When no reaction between phases occurs, the characteristic function evolves only by advection:

$$\partial_t \iota + \boldsymbol{v} \cdot \nabla \iota = 0 \tag{1.43}$$

To write a unique formulation in the whole domain Ω we express the density and the viscosity as the function of ι:

$$\rho = \iota\rho_1 + (1-\iota)\rho_2 \quad \text{and} \quad \mu = \iota\mu_1 + (1-\iota)\mu_2 \tag{1.44}$$

The conservation equation system on the whole domain Ω can be written as a function of the $d+2$ unknowns p, v_i and ι searched in their respective spaces $\mathcal{V}$, $\mathcal{P}$ and $\mathcal{C}$ that denote the restriction of Hilbert and Sobolev spaces with suitable boundary condition:

Find: $(\boldsymbol{v}, p, \iota) \in \mathcal{V} \times \mathcal{P} \times \mathcal{C}$, such that,

$$\begin{aligned} \partial_t \iota + \boldsymbol{v} \cdot \nabla \iota &= 0 \\ \rho\left(\partial_t \boldsymbol{v} + \boldsymbol{v} \cdot \nabla \boldsymbol{v}\right) + \nabla \cdot (p\mathbf{I} - 2\mu \mathbf{D}(\boldsymbol{v})) &= \sigma\kappa\delta_\Gamma \boldsymbol{n} + \rho \boldsymbol{g} \\ \nabla \cdot \boldsymbol{v} &= 0 \end{aligned} \tag{1.45}$$

Depicting the interface by a discontinuous characteristic function is not possible when using a continuum model for the two fluids. This function is replaced by the piecewise continuous function which has the following properties:

$$\iota(\boldsymbol{x},t) = \begin{cases} 1, & \text{for} \quad \boldsymbol{x} \in \Omega_1(t) \\ 0, & \text{for} \quad \boldsymbol{x} \in \Omega_2(t) \\ 0 < \iota(\boldsymbol{x},t) < 1, & \text{for} \quad \boldsymbol{x} \text{ in the transition area} \end{cases} \tag{1.46}$$

The smoothness of ι allows to compute directly the curvature κ of the interface. The gradient of ι gives a normal $\boldsymbol{n}$ that always point from fluid 2 ($\iota = 0$) to fluid 1 ($\iota = 1$):

$$\boldsymbol{n} = \nabla \iota \tag{1.47}$$

Indeed, as ι is constant in $\Omega_1(t)$ and $\Omega_2(t)$, the gradient is zero everywhere except in the transition area. From this normal $\boldsymbol{n}$ we can build the curvature such as::

$$\kappa = \nabla \cdot \left(\frac{\boldsymbol{n}}{\|\boldsymbol{n}\|} \right) \tag{1.48}$$

The discretization is handled by Finite Volume, and an explicit-implicit algorithm strategy is used to couple the description of the two phases and their evolution with the Navier-Stokes equations. The equation associated to the characteristic function is solved with an explicit time integration scheme whereas the other term can be solved with implicit time integration schemes. For more details on how the `OpenFOAM` CFD code, used herein as a component, handle this problem, see [Ubbink, 1997, Rusche, 2002].

1.3.2.3 *Dam break example*

The dam break example is a classic example. It is studied from the experimental [Bullock et al., 2007], the analytical [Stoker, 1992] as well as the numerically point of view [Ubbink, 1997, Dutykh and Mitsotakis, 2009], and describes the collapse of a water column hiting a rigid structure. This example is also proposed as a tutorial in `OpenFOAM` [OpenCFD LTD, 2009].

The geometry of the problem is exactly the same as the one taken in the last reference and can be seen in Fig. 1.10. The high density fluid representing water has the following density and kinematic viscosity: $\rho_1 = 1000 kg.m^{-3}$ and $\nu_1 = 1 \times 10^{-6} m.s^{-2}$. The low density fluid – air – is described with a density $\rho_2 = 1 kg.m^{-3}$ and a kinematic viscosity $\nu_2 = 1.48 \times 10^{-5} m.s^{-2}$. The flow in each phase is considered to be Newtonian (no turbulence effect is taken into account for instance). The surface tension between each phase is $0.07 kg.s^{-2}$, but regarding to the characteristic size of the problem, the surface tension is of little influence and can be neglected.

At $t = 0$, the initial condition are at rest, and the water column starts to collapse. Computation is run on window of size $\Delta t = 0.005$, but in each window, the time step for the flow time integration is dynamically determined in order to maintain the Courant–Friedrichs–Lewy number under 0.5. At the end of each window, the fluid force on the obstacle is obtained.

In Fig. 1.10, the evolution of the phases on two meshes are represented with respectively 2268 cells for the coarse mesh and 9072 cells for the fine mesh. The computation time are run on one $3.0 Ghz$ Intel Opteron processor. For the coarsest grid considered, no sub-iteration on the window is required to maintain the Courant number under the imposed limit. For the finest grid, some sub-iterations after the water column hit the rigid structure are required.

In Fig. 1.9 the forces acting upon the obstacle are also represented. We observe for the two discretizations a spurious point when, for the first time, a large air cavity is enclosed within a fluid tongue. However, regarding the value

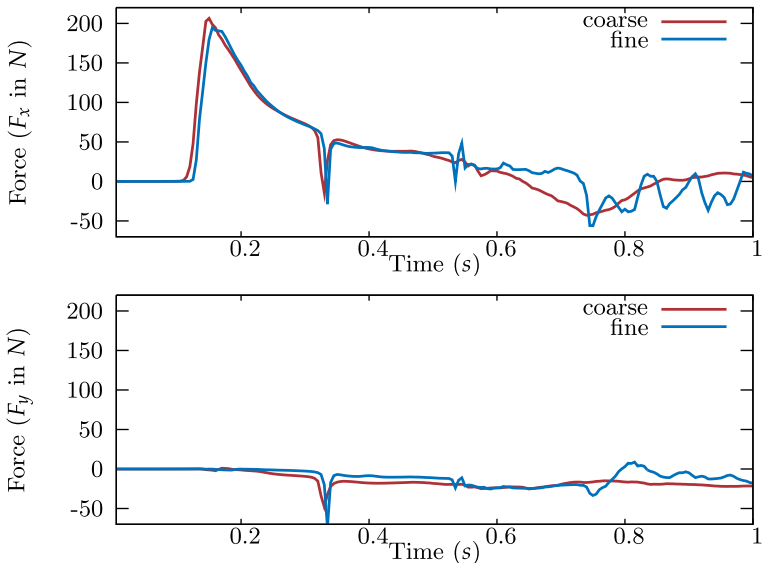


Figure 1.9: Horizontal and vertical forces acting upon the obstacle for coarse (2268 cells) and fine (9072 cells) grids

of the force along the $\boldsymbol{e}_x$ direction, the same behavior is observed before the tongue hits the right wall ($t \simeq 0.3s$). After that, the fine mesh is able to capture the separation of the water phase in many sub-domains and to represent more accurately their separated impact on the right side of the obstacle.

Closure

This first chapter was dedicated to the explanation of the solid and fluid strategies used in this dissertation. As one of the goals of this Ph. D. work is to emphasize the possibility to couple existing scientific computing methods and software to solve multiphysics problems, we use herein the widely known FE method for the structure and the FV method for the fluid. The way to solve the incompressibility constraint in a semi-implicit *segregated method* with the PISO algorithm is also described.

The FV capacity to solve the Navier-Stokes equation is illustrated by benchmark problems. Its extension to flows under moving domain, required for the field of application of fluid-structure interaction is presented herein. For moving domains where the shape remains regular an ALE framework is introduced. The mesh motion is insured by a smoothing operator based on a Laplacian operator. For moving domains with complicated shape motions such as fluid domain defined under a free-surface, a different approach is used. A biphasic modeling using a five equation model is introduced: velocity and pressure for the two phases as well as characteristic functions of the flow domain are solved with a *segregated method*. This strategy is applied to the fall of a water column.

The two subproblems for the fluid and solid being introduced, remains the question how to couple these problems. In the next chapter, the partitioned

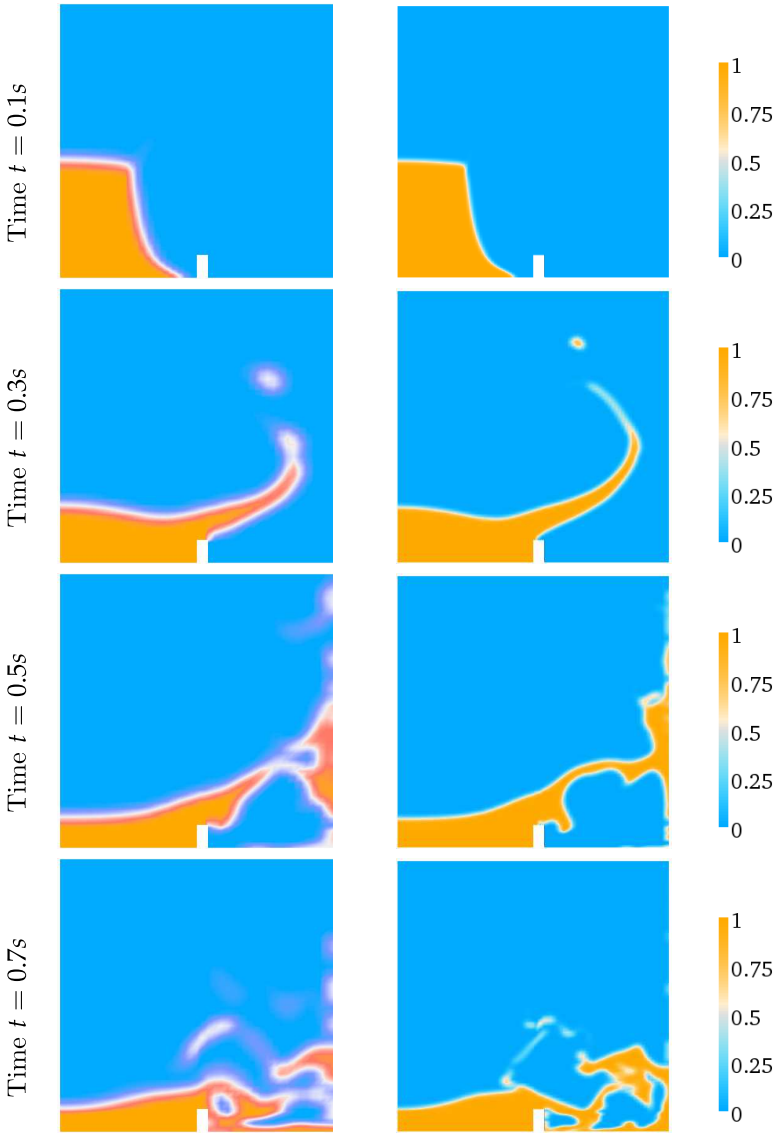


Figure 1.10: Evolution of the characteristic function ι depicting the density of each phases in a cell for fine and coarse meshes

approach used in this work is detailed.

2

Partitioned approach for fluid-structure interaction

In this chapter, different partitioned approaches to solve multiphysics problems – especially fluid-structure interaction – are presented. The way to handle coupled problems is first introduced in a general framework. Notations are set in Section 2.1.3. Explicit (Sec. 2.2) and Implicit coupling schemes (Sec. 2.3) based on Direct Force-Motion Transfer strategies are then studied.

Contents

2.1 Solving a coupled problem 36
2.1.1 Monolithic strategy 36
2.1.2 Partitioned strategy 37
2.1.3 Partitioned strategy notations 38
2.2 Explicit coupling 41
2.2.1 Generalized Conventional Serial Staggered algorithm 41
2.2.2 Evaluation of interface energy for DFMT-GCSS . . . 43
2.2.3 Enforcement of the Geometric Conservation Law . . 45
2.2.4 Improved Serial Staggered (ISS) algorithm 45
2.2.5 Conventional Parallel Staggered (CPS) algorithm . . 47
2.2.6 Explicit coupling with incompressible flows: the artificial "Added-Mass Effect" 47
2.3 Implicit coupling strategies 48
2.3.1 Algebraic solvers based on Picard iterations 49
2.3.2 Relaxation techniques 51
2.3.2.1 Fixed relaxation 51
2.3.2.2 Aitken's relaxation 52
2.3.2.3 Steepest descent technique 54
2.3.2.4 Relaxation seen as an approximative Newton algorithm 55
2.3.3 Newton and quasi-Newton based strategy 55
2.3.3.1 Formulation as a root equation 55
2.3.3.2 Generalized Minimum Residual Algorithm . 56
Closure 58

Solving multiphysics problems is a mainstream of current research in numerical method for engineering. As shown in Fig. 2.1, two approaches can be considered:

Monolithic strategy (see Section 2.1.1) where the coupled problem is stated in the same framework, and traditionally discretized with the same techniques. It leads to efficient solvers at the cost of a high development. Furthermore, as the cost to solve most of the algebraic systems obtained after discretization is non linear, this approach is not well suited for large scale problems.

Partitioned strategy (see Section 2.1.2) will try to keep the independence between solvers, and even full independence of their implementation in the corresponding software products. With this approach, each solver can be based on different discretization techniques (*e.g.* Finite Volume for the fluid in `OpenFOAM` [OpenCFD LTD, 2009] versus Finite Element for structure in `FEAP` [Taylor, 2008]) and employ different time-integration schemes and even different time steps.

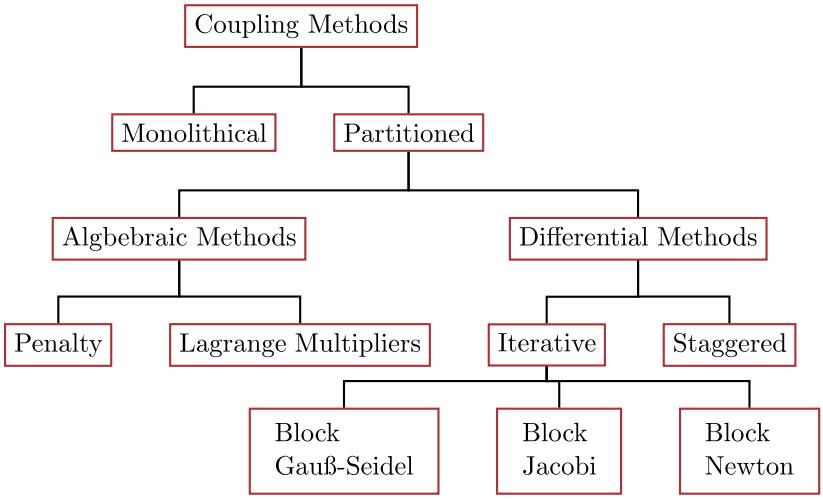


Figure 2.1: Main approaches to solve coupling problems [Jürgens, 2009]

The general coupling framework presented in Fig. 2.1, can be applied to fluid-structure interaction. In the following, the "Differential methods" are termed as Direct-Force Motion Transfer (DFMT) method, thus describing the data exchange between partitions. The "staggered coupling" where subproblems are solved one after another is here noted as explicit. The "iterative" coupling strategy solve implicitly the multiphysics problem.

In this work, the partitioned approach is chosen, as one of the goals of this work was to re-use existing software in a more general context. The methods developed for fluids and structures are carried out by different research teams,

based on different methods and solved by different tools, the partitioned approach here is a natural choice. Furthermore, considering realistic applications in fluid-structure interaction problems (namely, with turbulence) the time scales to accurately depict the fluid flow is much smaller than the one required for the solid part, which can be efficiently handled only by a partitioned approach that will synchronize the motion of both components at chosen time steps.

In this work, the partitioned approach is chosen, as one of the goals of this work was to re-use existing software in a more general context. Various techniques can then be applied to solve the Partitioned problem. They will be detailed in Sec. 2.2 and 2.3. The properties of the implemented algorithm (Chapter. 3) are mathematically studied in Appendix A.

2.1 Solving a coupled problem

2.1.1 *Monolithic strategy*

In the monolithic approach, the whole coupled problem is mathematically formulated in the same framework. For fluid structure interaction,this is, if not easily, generally well documented (See [Salençon, 2005] for a general introduction to continuum mechanics). For instance, the weak forms for the structure (see Sec. (1.1.2) and the fluid (see Sec. 1.3.1.1) subproblems can be summed up in order to obtain a weak form of the coupled problem (with appropriate continuity consideration at the interface). A numerical approach is then proposed to discretize this formulation in both space and time. It leads to an algebraic system that can be solved by any direct or iterative known algorithm. Traditionally, in this kind of approach, the numerical techniques proposed need to be the same for each subproblem. Considering that a formulation leads to the definition of two residuals r_f and r_s. A simplified form of the residual considered for the coupled problem can be written as:

$$\begin{cases} r_f(\boldsymbol{u}_f, \boldsymbol{u}_s) &= 0 \\ r_s(\boldsymbol{u}_f, \boldsymbol{u}_s) &= 0 \end{cases} \tag{2.1}$$

where $\boldsymbol{u}_f$ and $\boldsymbol{u}_s$ are the field describing the evolution of each subproblem. In a primal formulation for a structure, $\boldsymbol{u}_s$ is the displacement field, whereas $\boldsymbol{u}_f = (\boldsymbol{v}, p)$ takes into account both velocity and pressure for an incompressible flow.

Then, a Newton algorithm can be used to solve this non-linear residual problem:

$$\begin{bmatrix} \partial_{\boldsymbol{u}_f} r_f & \partial_{\boldsymbol{u}_s} r_f \\ \partial_{\boldsymbol{u}_f} r_s & \partial_{\boldsymbol{u}_s} r_s \end{bmatrix} \cdot \begin{bmatrix} \Delta \boldsymbol{u}_f \\ \Delta \boldsymbol{u}_s \end{bmatrix}^{(k)} = - \begin{bmatrix} r_f \\ r_s \end{bmatrix}^{(k)} \tag{2.2}$$

It is then required to access all the variables of each each subproblems, and to both diagonal and cross derivatives. The latter are not easy to obtain, and often necessitate numerical development in current software. Furthermore, the system obtained is large (sum of subproblem unknown) and necessitate to pay a high price for the storage and the inversion of the Jacobian. Thus, in the author's opinion, the advantages obtained in terms of order of convergence do not often counterbalance these drawbacks. For instance, as described

in [Ferziger and Perić, 2002] for incompressible fluid problems, computing the Jacobian is often so expensive that it is more advantageous to solve the coupled velocity-pressure problems with a fixed-point iterative strategy that does not rely on the exact computation of the tangent matrix. This remark applied to fluid alone remains true for fluid-structure interaction problems.

It is clear that the partitioned approach, introducing new unknowns at the interface, will lead to the storage of more variables. The size of the interface is one of the main criteria that determines the efficiency of a partitioned approach. For coupling problems (for instance thermo-mechanics, magneto-mechanisms,...) regarding a whole space-domain, one normally applies a monolithic approach [Felippa and Park, 2004]. Nevertheless, partitioned approach can be used for these coupled problems in the special case of different discretization either on the space or time domains [Kassiotis et al., 2009a].

In the context of fluid-structure interaction, noticeable work based on monolithical development can be found [Walhorn, 2002, Hübner et al., 2004]. All of these approaches use a Finite Element Formulation for fluid, solid and mesh motions. The Navier-Stokes equation is solved using a stabilized formulation [Brooks and Hughes, 1990, Franca et al., 1993]. For the time integration, either standard time integration techniques or Finite Elements in time can be used. In fluid-structure interaction the use of implicit generalized α-HHT method is recommended for its computational efficiency and its robustness, as studied in [Dettmer and Perić, 2003]. Some other works apply techniques arising from the research on computational fluid, and rely on the FV discretization of a problem formulated on an Eulerian grid [Mehl et al., 2008].

Note that a major trend in these works is to compute the Jacobian inexactly, and especially the cross terms of the tangent matrix one gets after linearization and discretization by FEM in [Perić et al., 2006, Dettmer and Perić, 2007]. Another way to improve efficiency comes at the cost of an elaborate coding adding with specially adapted "mixture" of explicit-implicit schemes as presented in [Belytschko et al., 1979, Hughes et al., 1979, Felippa and Park, 2004].

2.1.2 *Partitioned strategy*

When the methods developed for fluids and structures are carried out by different research teams, based on different methods and solved by different tools, the partitioned approach is a natural choice. Furthermore, considering realistic applications in civil engineering, the time scales to accurately depict the fluid flow is much smaller than the one required for the solid part.

The partitioned approach in fluid-structure interaction, first introduced in [Felippa et al., 1977] to model the impact of cavitating acoustic fluid on submarines, was mainly motivated by the lack of access to the code used to model one of the fields. Even in this early and in many later publications, the difficulty to keep a partitioned algorithm stable, even for stable and converging subproblems, was noticed. Other then time synchronization, another difficulty that arises from partitioned strategy is to transmit the most accurate information at the subproblems interface for matching and non-matching space discretization. In that respect, the partitioned methods can be divided into two large categories (see Fig. 2.1):

Differential methods – Direct Force Motion Transfer (DFMT);
: they were the first methods introduced to solve FSI in a partitioned manner. In DFMT, no additional unknowns are introduced at the interface, but the primal quantities (displacements or velocities) of one subproblem and the dual of the other one (pressures or forces) are exchanged in an explicit or implicit (iterative) way. The interface condition is however never strictly enforced until convergence of the methods. Then, the equality is fullfilled at a certain tolerance.

Algebraic methods – (Localized) Lagrange Multipliers (LLM);
: the general idea is to introduce a Lagrange multipliers field, in order to fulfill the algebraic equation at the interface. In the mortar method [Bernardi et al., 1990] the two subproblems are glued by one Lagrange multiplier field. The choice of the discretization of the Lagrange multiplier field can be a problem in itself. In the three-field method an additional displacement field of the fictitious interface is introduced, and two Lagrange multiplier fields glue each subproblem to the interface [Park et al., 1997].

LLM-based methods are known to give better convergence properties than the DFMT method, allowing, for instance, the use of bigger window size without loss of stability. However, in [Ross et al., 2009], a simplified comparison between LLM and DFMT shows that at each iteration, solving an FSI problem using the LLM is at least 4.5 times more expansive computationally speaking than solving it with DFMT. This is the case even for non-matching interface with a transfer operation of the interface field. The use of LLM is justified when one aims at solving problems with large time steps. Furthermore, as noticed in [Hautefeuille, 2009], the total equivalence between each subproblem allows the introduction of a multiscale framework in an elegant way.

In the next Section 2.1.3 the notation for the partitioned are given in order to get more into the details for the DMFT based methods.

2.1.3 *Partitioned strategy notations*

The interaction problem of motion of fluid (denoted further with f) and structure (denoted with s) is considered. For the sake of generality, it is supposed that theses problems can be, and in the general are, nonlinear and time dependent. It is also supposed that a Steklov-Poincaré operator can be built for each:

$$\begin{array}{rcl} \mathcal{S}_i : \mathcal{H}^{\frac{1}{2}}(\Gamma) & \rightarrow & \mathcal{H}^{-\frac{1}{2}}(\Gamma) \\ \boldsymbol{u}_i & \rightarrow & \boldsymbol{\lambda}_i \end{array} \tag{2.3}$$

that gives for an imposed primal quantity $\boldsymbol{u}_i$ defined on the coupling space and time domain $\Gamma \times [0, T]$, the evolution of its dual $\boldsymbol{\lambda}_i$.

For the problems considered in this work, the Steklov-Poincaré operators are not available analytically, but rather as the results of numerical approximations. Hence they naturally not only depend on the chosen model, material properties and boundary conditions, but also on discretization techniques, time integration algorithms, inverse discrete equations solvers...

The coupling considered in the following is based on two classic mechanics principles:

CONTINUITY OF PRIMAL QUANTITIES over the interface:

$$\boldsymbol{u}_s = \boldsymbol{u}_f = \boldsymbol{u} \qquad \text{on } \Gamma \times [0, T] \tag{2.4}$$

where $\boldsymbol{u}$ denotes the value of the primal variable at the interface. The latter can be interpreted as the perfect matching condition. Note that in the continuum space, the time derivatives of this condition give the equivalent equations for velocity $\boldsymbol{v} = \dot{\boldsymbol{u}}$:

$$\boldsymbol{v}_s = \boldsymbol{v}_f = \boldsymbol{v} \qquad \text{on } \Gamma \times [0, T] \tag{2.5}$$

and acceleration $\boldsymbol{a} = \dot{\boldsymbol{v}}$:

$$\boldsymbol{a}_s = \boldsymbol{a}_f = \boldsymbol{a} \qquad \text{on } \Gamma \times [0, T] \tag{2.6}$$

As a result of time discretization, these conditions are no longer equivalent.

EQUILIBRIUM OF DUAL QUANTITIES (action-reaction principle): which implies equilibrium of dual variables (stress) on the interface:

$$\boldsymbol{\lambda}_f + \boldsymbol{\lambda}_s = \boldsymbol{0} \qquad \text{on } \Gamma \times [0, T] \tag{2.7}$$

The action-reaction principle (Eq. 2.7) can be reformulated using the Steklov-Poincaré operator defined in Eq. (2.3). Thus, the solution of the coupled problem can be given as the so-called Steklov-Poincaré formulation:

Find: $\boldsymbol{u}$ on $\Gamma \times [0, T]$,
So that:

$$\mathcal{S}_f(\boldsymbol{u}) + \mathcal{S}_s(\boldsymbol{u}) = \boldsymbol{0} \tag{2.8}$$

Using the inverse of the first Steklov-Poincaré operators allows to re-write the equilibrium of dual quantities as the following fixed-point equation that concerns only the unknown at the interface:

Find: $\boldsymbol{u}$ on $\Gamma \times [0, T]$,
So that:

$$\boldsymbol{u} = {\mathcal{S}_s}^{-1}\left(-\mathcal{S}_f(\boldsymbol{u})\right) \tag{2.9}$$

It is recalled that the Steklov-Poincaré operator $\mathcal{S}$ is *a priori* as expensive to compute as its associated inverse $\mathcal{S}^{-1}$, but not more.

The fixed-point equation can be reformulated in order to get a root equation:

Find: $\boldsymbol{u}$ on $\Gamma \times [0, T]$,
So that:

$$ {\mathcal{S}_s}^{-1}\left(-\mathcal{S}_f(\boldsymbol{u})\right) - \boldsymbol{u} = \boldsymbol{0} \tag{2.10}$$

For the three formulations, the quantities requested are defined at the interface of the problem and demand the proper definition of the Steklov-Poincaré or Poincaré-Steklov operators for the fluid and the structure.

Algorithm 3 Steklov-Poincaré operator for fluid

Require: Fluid state variable at time T_N

1: Impose displacement of mesh at fluid-structure interface:
$$\boldsymbol{u}_f = \boldsymbol{u} \text{ on } \Gamma \times [T_N, T_{N+1}]$$
2: Solve mesh internal nodes displacement:
$$\mathcal{R}_m(\boldsymbol{u}_f; \boldsymbol{u}) = 0 \text{ on } \Omega_f \times [T_N, T_{N+1}]$$
3: Solve fluid problem in ALE formulation:
$$\mathcal{R}_f(\boldsymbol{v}_f, p_f; \boldsymbol{u}_f) = 0 \text{ on } \Omega_f \times [T_N, T_{N+1}]$$
4: Get boundary traction force at the interface:
$$\boldsymbol{\lambda} = -\boldsymbol{\sigma}_f \boldsymbol{n} \text{ on } \Gamma \times [T_N, T_{N+1}]$$

Algorithm 4 Poincaré-Steklov operator for structure

Require: Solid state variable at time T_N

1: Impose boundary traction force at fluid-structure interface:
$$\boldsymbol{\sigma}_s \boldsymbol{n} = \boldsymbol{\lambda} \text{ on } \Gamma \times [T_N, T_{N+1}]$$
2: Solve structure problem:
$$\mathcal{R}_s(\boldsymbol{u}_s; \boldsymbol{\lambda}) = 0 \text{ on } \Omega_s \times [T_N, T_{N+1}]$$
3: Get boundary displacement at the interface:
$$\boldsymbol{u} = \boldsymbol{u}_s \text{ on } \Gamma \times [T_N, T_{N+1}]$$

However, even if Steklov-Poincaré operators $\mathcal{S}$ are expressed only at the interfaces, their computation concerns the whole problem on the sub-domains which makes them quite expensive.

Furthermore, interface equilibrium equation is nonlinear, and an iterative scheme has to be defined to compute its solution or at least a good approximation. For each of the formulation defined in Eq. (2.8)–(2.10), an associated strategy is naturally defined:

Fixed-point formulation Eq. (2.9): the easiest way to solve this equation is to consider Picard like iterative schemes (Section 2.3.1), whose stability and convergence are conditional [Wall, 1999, Le Tallec and Mouro, 2001]. When they are not converging, the fixed-point equation Eq. (2.9) can be re-written as a nonlinear root equation.

Root problem formulation Eq. (2.10): is the formulation to choose to introduce in a natural way Newton algorithms [Matthies and Steindorf, 2003, Fernández and Moubachir, 2005] or its approximation by quasi-Newton techniques [Gerbeau and Vidrascu, 2003a]; The computation of the Jacobian, and especially of its crossed terms, can be quite expensive, and therefore the use of quasi-Newton techniques is often recommended.

Steklov-Poincaré formulation Eq. (2.8): is the choice when domain decomposition techniques are applied to the fluid-structure interaction problem. Following the presentation made in [Deparis, 2004], the non-linear

Richardson strategy can be applied to solve the interface equation of the coupled problem [Deparis et al., 2006, Fernández et al., 2008].

REMARK: Regarding discrete continuity condition, authors do not always agree on whether the displacement or the velocity continuity should rather be enforced. Generally, the continuity of displacements is seen a better choice for the ALE formulation for the fluid subproblem since the interface motion (imposing the mesh motion) is already described by Lagrangian equations.

2.2 Explicit coupling

Sometimes termed as the staggered approach it aims at getting a good approximation of the coupled problem solving only one Steklov-Poincaré operator by time step and by field.

2.2.1 *Generalized Conventional Serial Staggered (GCSS) algorithm*

Consider the evolution of the coupled problem in a window $[T_N, T_{N+1}]$ of size Δt. The idea is to solve at each time step a single Picard-iteration of the fixed-point equation in Eq. (2.9):

$$\boldsymbol{u}_{N+1} = {\mathcal{S}_s}^{-1}\left(-\mathcal{S}_f\left(\boldsymbol{u}_N\right)\right) \tag{2.11}$$

However, this method has bad conservation properties at the interface. These properties can be improved by the addition of a predictor $\mathcal{P}$ for the interface displacements which results with the following algorithm:

Algorithm 5 Generalized Conventional Serial Staggered

1: Given: initial counter $N = 0$, initial time $T = T_0$, final time T_{max}, window size Δt, initial interface displacement $\boldsymbol{u}_0$.
2: **while** $T < T_{\text{max}}$ **do**
3: Predict displacement: $\boldsymbol{u}^{\mathcal{P}}_{N+1} = \mathcal{P}(\boldsymbol{u}_N, \dot{\boldsymbol{u}}_N, \boldsymbol{u}_{N-1}, \dots)$
4: Solve problem f: $\boldsymbol{\lambda}_{N+1} = \mathcal{S}_f(\boldsymbol{u}^{\mathcal{P}}_{N+1})$
5: Solve problem s: $\boldsymbol{u}_{N+1} = {\mathcal{S}_s}^{-1}(\boldsymbol{\lambda}_{N+1})$
6: $N \leftarrow N + 1$ and $T \leftarrow T_0 + N \times \Delta t$
7: **end while**

Note that we do not impose the way the time integration is performed, but only that the fluid and the structure part be collocated at the end of the same time windows. When considering equal time step size for fluid and structure, this Direct Force-Motion Transfer algorithm is named Conventional Serial Staggered (DFMT-CSS) (see [Farhat et al., 1995]). We can also consider the so-called Subcycled Conventional Staggered Scheme (DFMT-SCSS) with smaller time steps selected for integration of one of the subproblems. Usually, the characteristic time scale of the fluid is smaller, and it is natural to consider its integration with many small time steps on the window $[T_N, T_{N+1}]$ for the fluid part.

The displacements $\boldsymbol{u}$ seen by the structure subproblem s at time T_{N+1} is therefore $\boldsymbol{u}_{N+1}$ whereas it remains a direct function of $\boldsymbol{u}_N$ for the fluid subproblem f. For that reason, the interface condition at T_{N+1} has no reason to be

fulfilled for the velocity:

$$\boldsymbol{u}_{s,N+1} - \boldsymbol{u}_{f,N+1} = \boldsymbol{u}_{N+1} - \boldsymbol{u}^{\mathcal{P}}_{N+1} \neq 0 \tag{2.12}$$

Therefore, an energy error is introduced by the exchanges at the interface with this kind of algorithm. This energy transfer error can be estimated by computing the energy seen at the interface for each subproblem [Piperno, 2000].

For a fluid point located at the interface ($\Omega_f \cap \Gamma$), the energy estimate takes the following general form (this formula is exact in one-dimension):

$$\Delta e_{f,N+1} = -\boldsymbol{\lambda}^{\star}_f \cdot \left(\boldsymbol{u}_{f,N+1} - \boldsymbol{u}_{f,N}\right) = -\boldsymbol{\lambda}^{\star}_f \cdot \left(\boldsymbol{u}^{\mathcal{P}}_{N+1} - \boldsymbol{u}_N\right) \tag{2.13}$$

where $\boldsymbol{\lambda}^{\star}_f$ depends on the algorithm chosen for the time integration of the fluid problem. For instance, this dual quantity takes the following value for:

First-order forward-Euler explicit integration scheme,

$$\boldsymbol{\lambda}^{\star}_f = \boldsymbol{\lambda}_{f,N} \tag{2.14a}$$

First-order backward-Euler implicit integration scheme,

$$\boldsymbol{\lambda}^{\star}_f = \boldsymbol{\lambda}_{f,N+1} \tag{2.14b}$$

Second order time integration, a good approximation is given by:

$$\boldsymbol{\lambda}^{\star}_f \simeq \frac{\boldsymbol{\lambda}_{f,N} + \boldsymbol{\lambda}_{f,N+1}}{2} \tag{2.14c}$$

Fluid sub-cycling on the window $[T_N, T_{N+1}]$:

$$\boldsymbol{\lambda}^{\star}_f \simeq \frac{1}{\Delta t} \int_{T_N}^{T_{N+1}} \boldsymbol{\lambda}_f(t) \mathrm{d}t \tag{2.14d}$$

When sub-cycling is applied, the energy estimate given by the formula stated in Eq. (2.13) assumes that fluid interface velocity is constant on the window $[T_N, T_{N+1}]$ and given by Δt. Hence, the value of the dual at the interface after the fluid computation $\boldsymbol{\lambda}_{N+1}$ can take any of the values stated above.

Reciprocally, for a point of the structure at the interface $\Omega_s \cap \Gamma$, the energy transfered by the fluid is estimated with:

$$\Delta e_{s,N+1} = \boldsymbol{\lambda}^{\star}_s \cdot (\boldsymbol{u}_{s,N+1} - \boldsymbol{u}_{s,N}) = \boldsymbol{\lambda}^{\star}_s \cdot (\boldsymbol{u}_{N+1} - \boldsymbol{u}_N) \tag{2.15}$$

where $\boldsymbol{\lambda}^{\star}_s$ depends also on the time integration scheme for the structure part. For all kinds of schemes applied to the fluid domain described from Eq. (2.14a) to (2.14d), the same results hold for the structure part. We consider here only second order schemes that lead to:

$$\boldsymbol{\lambda}^{\star}_s \simeq \frac{\boldsymbol{\lambda}_{s,N} + \boldsymbol{\lambda}_{s,N+1}}{2} \tag{2.16}$$

So, the behavior of coupling schemes is influenced by the way one computes the flow induced load on the structure. It is natural to consider the following choices proposed in [Piperno and Farhat, 2001]:

$$\boldsymbol{\lambda}_{s,N+1} = \boldsymbol{\lambda}_N \tag{2.17a}$$
$$\boldsymbol{\lambda}_{s,N+1} = \boldsymbol{\lambda}_{N+1} \tag{2.17b}$$
$$\boldsymbol{\lambda}_{s,N+1} = \frac{\boldsymbol{\lambda}_{N+1} + \boldsymbol{\lambda}_N}{2} \tag{2.17c}$$
$$\boldsymbol{\lambda}_{s,N+1} = \frac{1}{\Delta t} \int_{T_N}^{T_{N+1}} \boldsymbol{\lambda}(t) \mathrm{d}t \tag{2.17d}$$
$$\boldsymbol{\lambda}_{s,N+1} = 2\boldsymbol{\lambda}_N - \boldsymbol{\lambda}_{s,N} \tag{2.17e}$$
$$\boldsymbol{\lambda}_{s,N+1} = 2\boldsymbol{\lambda}_{N+1} - \boldsymbol{\lambda}_{s,N} \tag{2.17f}$$
$$\boldsymbol{\lambda}_{s,N+1} = \frac{2}{\Delta t} \int_{T_N}^{T_{N+1}} \boldsymbol{\lambda}(t) \mathrm{d}t - \boldsymbol{\lambda}_{s,N} \tag{2.17g}$$

Clearly, as stated in Eq. (2.12) for the displacements, the exchanged energy in Eq. (2.13) and Eq. (2.15) cannot exactly compensate each other as the predictor $\boldsymbol{u}_{N+1}^{\mathcal{P}}$ has no reason to be equal to the exact displacement $\boldsymbol{u}_{N+1}$. This leads to the following energy error at the interface:

$$\Delta e_{N+1} = \Delta e_{f,N+1} + \Delta e_{s,N+1} \tag{2.18}$$

Thus energy is created or damped at the interface of the fluid-structure interaction. This can deteriorate the accuracy expected from the integration schemes chosen for the subproblems, and yield a poor estimation of the fluid-structure interaction phenomena by numerical computation. On the one hand, the creation of energy at the interface will lead to an unstable algorithm. Indeed, if each of the subproblems considered is unconditionally stable, it is known that the explicit coupling approach is conditionally stable. The stability of the coupled problem is governed by the production of energy at the interface. On the other hand, when energy produced is negative, it can overdamp the problem, leading to an underestimation of the domain in which the coupling is crucial.

2.2.2 *Evaluation of interface energy for DFMT-GCSS*

In [Piperno and Farhat, 1997] an estimate of the energy error Δe for most of the fluid and structure time integration schemes and a predictor $\mathcal{P}$ is given. The main results obtained are hereby recalled. The author extends the mathematical study for a one-dimension oscillating piston problem proposed in [Piperno, 1995].

An asymptotic development for small window size Δt shows that the energy transfered after N time steps is proportional to

$$\sum_{i=1}^{N} \Delta e_i \sim \delta e \times N \quad \text{for small window size: } \Delta t \tag{2.19}$$

In the following, an explicit coupling scheme is of order n if $\delta e \sim C(\Delta t)^n$ when $\Delta t \longrightarrow 0$. The higher n is, the less energy produced or damped at the interface.

In the coupling component (Chapter 3) a component of the following form is implemented:

$$\boldsymbol{u}_{N+1}^{\mathcal{P}} = \mathcal{P}(\boldsymbol{u}_N, \dot{\boldsymbol{u}}_N, \dot{\boldsymbol{u}}_{N-1}) = \boldsymbol{u}_N + \alpha_0 \Delta t \dot{\boldsymbol{u}}_N + \alpha_1 \Delta t \left(\dot{\boldsymbol{u}}_N - \dot{\boldsymbol{u}}_{N-1}\right) \tag{2.20}$$

The discussion on the order of the coupling procedure is now stated as the function of the order of the predictor.

CONSISTENT PREDICTOR for $\alpha_0 = 0$ and $\alpha_1 = 0$. The coupling algorithm is then first order for most of the fluid integration schemes and input interface forces. The only way not to get a first order scheme is to consider the Euler explicit integration scheme for the fluid (see Eq. (2.14a)) and no correction of upload force on the solid (see Eq. (2.17a)). In that case, the energy error is at least of fourth order. But this algorithm is very stringent, as explicit computing in fluids has to verify the CFL condition. Furthermore, even if the energy error at the interface is small, the momentum equation is not verified well.

FIRST ORDER PREDICTOR for $\alpha_0 = 1$ and $\alpha_1 = 0$. The algorithm is proved to be second order accurate for most of integration schemes in the subproblems. For most of the cases, this leads to

$$\delta e = -\frac{1}{2} C_d \left(\frac{\Delta t}{T_{\text{char}}}\right)^2 + \mathcal{O}\left(\left(\frac{\Delta t}{T_{\text{char}}}\right)^3\right) \tag{2.21}$$

where C_d is a constant which depends on the problem treated, and represents added damping effects due to the coupling problem. An optimum error energy can however be obtained with:

$$\delta e = -\frac{5}{12} C_d \left(\frac{\Delta t}{T_{\text{char}}}\right)^2 + \mathcal{O}\left(\left(\frac{\Delta t}{T_{\text{char}}}\right)^3\right) \tag{2.22}$$

This smallest energy error at the interface is obtained for second order time-accurate flow integration $\boldsymbol{\lambda}_f^\star \simeq \frac{\boldsymbol{\lambda}_{f,N} + \boldsymbol{\lambda}_{f,N+1}}{2}$ and averaged induced load $\frac{2}{\Delta t}\int_{T_N}^{T_{N+1}} \boldsymbol{\lambda}(t)\mathrm{d}t - \boldsymbol{\lambda}_{s,N}$. For that case, the energy error is found to be $16,6\%$ smaller than when the force term is considered to be the same as in the fluid computation.

SECOND ORDER PREDICTOR for $\alpha_0 = 1$ and $\alpha_1 = \frac{1}{2}$. The highest order for expected is the third one, and is obtained when considering $\boldsymbol{\lambda}_{f,N+1} = \boldsymbol{\lambda}_{s,N+1}$, whatever the expression, that depends of the flow time integrator, of $\boldsymbol{\lambda}_{s,N+1}$. The momentum is then conserved and the energy error is governed by:

$$\delta e = \frac{5}{12} C_k \left(\frac{\Delta t}{T_{\text{char}}}\right)^3 + \mathcal{O}\left(\left(\frac{\Delta t}{T_{\text{char}}}\right)^4\right) \tag{2.23}$$

where C_k is a constant that depends on the problem treated, and represents the effects of stiffness and added mass.

2.2.3 *Enforcement of the Geometric Conservation Law (GCL)*

In order to be mathematically consistent, the computational method for the flow has to predict exactly a uniform flow on a moving grid. It was proved that the velocity of the dynamic mesh has to be computed for all first– and second–order time accurate methods as:

$$\dot{u}_f = \frac{u_{f,N+1} - u_{f,N}}{\Delta t} \tag{2.24}$$

This condition can be found in the literature for ALE formulations discretized either by Finite Volume [Demirdžić and Perić, 1988] or Stabilized Finite Element methods [Förster et al., 2006]. It is usually termed Geometric Conservation Law (GCL) and sometimes the Space Conservation Law (SCL). In [Farhat et al., 2001], the advantages in strictly enforcing the GCL for any flow computation on a moving domain are recalled. GCL enforcing on the specific contexts of FSI was introduced in [Lesoinne and Farhat, 1996]. Thus, the use of an algorithm that enforces the GCL allows for instance the use of a time step ten times bigger than its inappropriate counterpart [Farhat and Lesoinne, 2000]. Even worse, error in the GCL can lead to an underestimation of the flutter speed in aeroelastic problems.

The previous DFMT-GCSS algorithm was collocated. It means that the window for computation on the fluid and solid parts were the same: $[T_N, T_{N+1}]$. It implies that both continuity and Geometric Conservation Law (GCL) cannot be satisfied simultaneously. Indeed, even for a perfect prediction $u^{\mathcal{P}}_{N+1} = u_{N+1}$ that has really little chance to occur, the velocity of the mesh on the fluid side is computed as such:

$$\dot{u}_f = \frac{u_{f,N+1} - u_{f,N}}{\Delta t} = \frac{u_{N+1} - u_N}{\Delta t} \tag{2.25}$$

and cannot be equal to the velocity on the solid side, since for second order schemes such as the midpoint or the trapezoidal rule, velocity is computed as:

$$\frac{u_{N+1} - u_N}{\Delta t} = \frac{u_{f,N+1} - u_{f,N}}{\Delta t} \neq \dot{u}_s \tag{2.26}$$

In order to be accurate, the DFMT-GCSS has to be used with small time steps.

2.2.4 *Improved Serial Staggered (ISS) algorithm*

To improve the explicit algorithm presented above, a natural idea is to consider a non-collocated algorithm where the displacements $\boldsymbol{u}$ and the forces $\boldsymbol{\lambda}$ are not computed at the same moment. This idea is summarized in **Algorithm 6**.

As shown in [Farhat and Lesoinne, 2000], this algorithm enforces both the continuity equation (Eq. (2.4)) and the GCL (Eq. (2.24)) for second order time integration schemes on the solid part. The idea behind this is to consider the following equality:

$$\begin{aligned} \dot{u}_{f,N} &= \frac{u_{f,N+\frac{1}{2}} - u_{f,N-\frac{1}{2}}}{\Delta t} \\ &= \frac{1}{\Delta t}\left(u_{s,N} + \frac{\Delta t}{2}\dot{u}_{s,N} - u_{s,N-1} - \frac{\Delta t}{2}\dot{u}_{s,N-1}\right) \end{aligned} \tag{2.27}$$

Algorithm 6 Improved Serial Staggered

1: Given: initial counter $N = 0$, initial time $T = T_0$, final time $T_{\max}$, window size Δt, initial interface displacement $\boldsymbol{u}_0$ and velocity $\dot{\boldsymbol{u}}_0$.
2: Compute: initial fluid mesh with: $\boldsymbol{u}_{-\frac{1}{2}} = \boldsymbol{u}_0 - \frac{\Delta t}{2}\dot{\boldsymbol{u}}_0$
3: **while** $T < T_{\max}$ **do**
4: Predict displacement: $\boldsymbol{u}^{\mathcal{P}}_{N+\frac{1}{2}} = \boldsymbol{u}_N + \frac{\Delta t}{2}\dot{\boldsymbol{u}}_N$
5: Solve problem f: $\boldsymbol{\lambda}_{N+\frac{1}{2}} = \mathcal{S}_f(\boldsymbol{u}^{\mathcal{P}}_{N+\frac{1}{2}})$
6: Solve problem s: $\boldsymbol{u}_{N+1} = \mathcal{S}_s{}^{-1}(\boldsymbol{\lambda}_{N+\frac{1}{2}})$
7: $N \leftarrow N+1$ and $T \leftarrow T_0 + N \times \Delta t$
8: **end while**

and to introduce the midpoint rule that states for the solid part:

$$u_{s,N} = u_{s,N-1} + \frac{\Delta t}{2}\left(\dot{u}_{s,N} + \dot{u}_{s,N-1}\right) \tag{2.28}$$

It leads to:

$$\dot{u}_{f,N} = \dot{u}_{s,N} \tag{2.29}$$

Note that the same kind of remarks holds when coupling together nonlinear structures. In [Ibrahimbegović and Mamouri, 2002], an algorithm that dissipate spurious high frequency for geometrically exact coupled beams is proposed. In [Piperno and Farhat, 2001], the interface energy produced by the DFMT-ISS is shown to be potentially of second or third order.

Second order accuracy coupling schemes are obtained for many combinations of the flow integrator and the evaluation of force. The most efficient second order DFMT-ISS is obtained for highly subcycled flows (Eq. (2.14d)) and corrected forces evaluated in the last instant of the flow integrator (Eq. (2.17e)). In this case, the energy produced at the interface can be approximated with:

$$\delta e = -\frac{1}{12}C_d\left(\frac{\Delta t}{T_{\text{char}}}\right)^2 + \mathcal{O}\left(\left(\frac{\Delta t}{T_{\text{char}}}\right)^3\right) \tag{2.30}$$

One can notice that the coefficient behind the second order term is smaller than for any collocated coupling scheme.

Third order accuracy coupling schemes are obtained by considering second order flow integrators without any sub-cycling (Eq. (2.14c)), and applying a corrected force evaluated at the end of the fluid window $T_{N+\frac{1}{2}}$ (Eq. (2.17e)). This is the efficient scheme proposed in [Farhat et al., 1995].

For both second- and third-order accuracy schemes the attention of the reader is focused the size of the coefficient behind the term of lowest order. Compared to the collocated scheme like DFMT-GCSS, the non-collocated coupling scheme DFMT-ISS exhibits smaller coefficients that explains the better properties observed for this scheme.

2.2.5 *Conventional Parallel Staggered (CPS) algorithm*

The algorithms proposed are naturally not parallel, in the sense that the structure problem cannot be solved on a window without knowing the computed evolution of the fluid that the same window. In FSI, this is sometimes proposed in order to gain time computation, but often yields interface.

However, in most computations, the time spent for the fluid solver is much longer that the one spent for the solid one. It seems more relevant to propose to take advantage of the fluid solver parallelization. More time was dedicated to the proposition of a parallel version of our fluid components (see Section 3.3.3 for more details) than a deep study of the DFMT-CPS algorithm and their implicit extensions.

2.2.6 *Explicit coupling with incompressible flows: the artificial "Added-Mass Effect"*

In the first analysis of explicit DFMT partitioned algorithms applied to fluid-structure interaction [Felippa et al., 1977], an upper limit on the time step size is obtained for which each numerical simulation of structure explicitly coupled to an acoustic flow diverges. The maximum time step size reported depends on the fluid/structure density ratio and on the speed of sound in the fluid medium. Such a criterion directly applied to incompressible flow where exist infinite wave speed, predicts immediate instability, whatever the coupling time step chosen.

This effect is termed artificial "Added-Mass Effect", since major parts of the fluid act as an extra mass moving with the structure [Le Tallec and Mouro, 2001]. In staggered schemes, the computed fluid forces acting on the structure depend on the predicted displacements instead of the exact ones which eventually lead to instabilities observed for all explicit coupling schemes for incompressible flows and structures. Those instability phenomenon are mentioned in [Wall, 1999, Le Tallec and Mouro, 2001, Mok, 2001]. In [Causin et al., 2005], a simplified model is proposed in order to predicts the instabilities. In [Förster et al., 2007, Förster, 2007], an added discrete mass operator is given in terms of matrices obtained with stabilized FEM discretization for the fluid part.

The added mass effect depends on some parameters of the discrete fluid-structure interaction problem. The influence of some of those parameters is now established, the influence of some parameters is as follows:

Coupling time step size: the smaller the time step is, the earlier the instability occurs;

Mass ratio between fluid and solid: the larger ρ_f/ρ_s, the earlier the instability occurs;

Temporal discretization precision order: the higher order the time integration scheme is, the earlier the instability occurs;

Fluid velocity: higher velocity has been numerically experimented to increase the instability;

Solid stiffness: stiff structure leads to better stability properties than soft ones.

The instability of explicit coupling algorithm is here emphasized by the numerical examples in Sec. 4.1 and 4.3.

2.3 Implicit coupling strategies

For some fluid-structure interaction problems, and especially for aero-elastic problems at high Mach number (compresssible flows), the explicit coupling strategies detailed in Sec. 2.2 are widely used. Such strategies, where the results from one computation are given to the next one without any "come-back", favor of simplicity small cost, but even if the error propagation at each data exchange can be estimated, it can lead to the over- or underestimation of physical instability phenomena. For coupled problems with incompressible flows, strictly enforcing the interface continuity is often much more stringent in order to avoid the added mass effect as explained in Sec. 2.2.6. By enforcing the continuity of primal variables at the interface the energy error can be eliminated (see Eq. (2.12)). This can be done by iterating on the following residual hoping to reduce it below a chosen tolerance:

$$\boldsymbol{r}_{N+1} := \boldsymbol{u}_{s,N+1} - \boldsymbol{u}_{f,N+1} \simeq 0 \quad \leq \texttt{TOL} \tag{2.31}$$

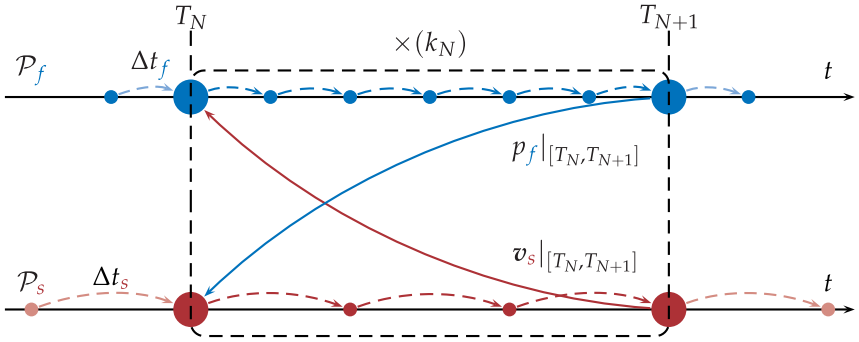


Figure 2.2: Block Gauß-Seidel coupling algorithm for fluid ($\mathcal{P}_f$) structure ($\mathcal{P}_s$) interaction problems; this iterative scheme is applied (k_N) times until convergence on a window $[T_N, T_{N+1}]$.

In this way we recover an implicit coupling solution. For each implicit coupling algorithm the main questions to address are:

- which data (physical quantities) to exchange, and in which order? In Fig. 2.2 the Block-Gauß-Seidel algorithm for fluid-structure interaction problem is presented.
- is the coupling algorithm stable? In fact, even if each subproblem is computed by a stable and converging scheme, a coupling algorithm can still diverge [Matthies et al., 2006, Arnold and Gunther, 2001].

Fig. 2.2 gives an illustration of the results obtained by independent time integration solvers in a window (*i.e.* $t \in [T_N, T_{N+1}]$), where not only the value

at synchronization points T_N or T_{N+1}, but also the interpolated evolution of considered variables on the whole window have to be exchanged. Note that one of the great advantages of the proposed algorithms is the possibility to use different time steps and integration schemes for the two subproblems considered. For example, the *segregated approach* with a PISO scheme in the VF-based code `OpenFOAM` can be chosen, and is only half-implicit and requires small time step in order to fulfill the CFL stability condition, along with an implicit scheme for the structure that allow bigger time steps. It is possible to run a coupled simulation that takes advantage of the natural time stepping arising from the fluid and solid subproblems and their discretization.

2.3.1 *Algebraic solvers based on Picard iterations*

Contrary to explicit coupling algorithms which introduce spurious energy at the interface, implicit ones will thus enforce the same evolution of the primal variable at the fluid-structure interface. This is carried out by solving iteratively the fixed point equation in (2.9) by the Picard iterative strategy like, which reads:

$$\boldsymbol{u}_{N+1}^{(k+1)} = \mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right); \qquad \mathcal{G} = {\mathcal{S}_s}^{-1} \circ -\mathcal{S}_f \tag{2.32}$$

where the fixed-point function is based upon the Steklov-Poincaré operators defined previously. The Picard iterations are carried out until convergence of interface residual is achieved:

$$\boldsymbol{r}_{N+1}^{(k)} = \boldsymbol{u}_{s,N+1}^{(k)} - \boldsymbol{u}_{f,N+1}^{(k)} = \mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right) - \boldsymbol{u}_{N+1}^{(k)} \tag{2.33}$$

Such a Picard iterative strategy can be recognized as the Block-Gauß-Seidel algorithm, further denoted as DFTM-BGS, which is a natural generalization of the explicit algorithms depicted above; It can be formally written as in **Algorithm 7**.

Algorithm 7 Direct Force-Motion Transfer Block-Gauß-Seidel

1: Given: initial counter $N = 0$, initial time $T = T_0$, final time $T_{\max}$, window size Δt, initial interface displacement $\boldsymbol{u}_0$.
2: **while** $T < T_{\max}$ **do**
3: $(k) = 0$
4: Predict displacement: $\boldsymbol{u}_{N+1}^{(0)} = \mathcal{P}(\boldsymbol{u}_N^{(k_{\max})}, \boldsymbol{u}_N^{(k_{\max})}, \boldsymbol{u}_{N-1}^{(k_{\max})}, \dots)$
5: **repeat**
6: Perform a Picard iteration: $\mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right)$
7: Compute residual: $\boldsymbol{r}_{N+1}^{(k)} = \mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right) - \boldsymbol{u}_{N+1}^{(k)}$
8: Update interface primal variable: $\boldsymbol{u}_{N+1}^{(k+1)} = \boldsymbol{u}_{N+1}^{(k)} + \boldsymbol{r}_{N+1}^{(k)}$
9: **do** $(k) \leftarrow (k) + 1$
10: **until** $\|\boldsymbol{r}_N^{(k-1)}\| \geq$ `TOL`
11: $N \leftarrow N + 1$ and $T \leftarrow T + \Delta t$
12: **end while**

The fixed-point algorithm based on Picard iterations for the time step $N+1$ is represented in Fig. 2.3. Here, the main drawback of this simple algorithm is

that the search direction for $\boldsymbol{u}$ and $\boldsymbol{\lambda}$ variables does not exploit any information from the fixed-point function $\mathcal{G}$ and the Steklov-Poincaré operators $\mathcal{S}_f$ and $\mathcal{S}_s$ that compose it. Therefore, many iterations can be required to reach the convergence and even sometimes the algorithm can show unstable behavior.

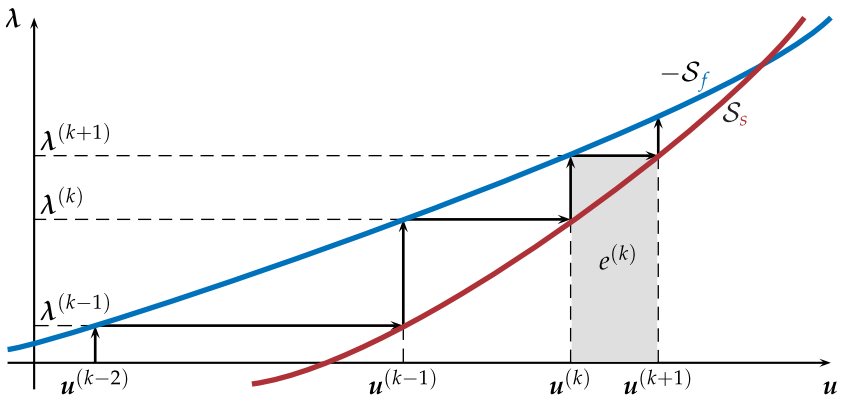


Figure 2.3: Strong Coupling algorithm by non-linear Block Gauß-Seidel.

The convergence of the coupling algorithm can be carried out using amplification matrices, and are therefore based on linear stability [Felippa and Park, 2004]. In Appendix A, a non-linear stability proof using the Differential Algebraic framework is proposed [Arnold and Gunther, 2001].

To improve the convergence and stability behavior of the DFMT-BGS method we can try a more accurate update:

$$\boldsymbol{u}_{N+1}^{(k+1)} = \boldsymbol{u}_{N+1}^{(k)} + \mathcal{H}\, \boldsymbol{r}_{N+1}^{(k)} \tag{2.34}$$

where $\mathcal{H}$ is a matrix that improves the method convergence. Such a matrix can be built with different methods:

Secant methods are used when the number of iterations is not the major issue, but rather trying to maintain the cost of each iteration as low as possible.

- traditional Block-Gauß-Seidel algorithm considers $\mathcal{H}$ to be the identity matrix. No more inverse at the interface level is required to compute the increment. However, the convergence rate is low and the stability domain of this algorithm is limited, especially when coupling incompressible flows and structures.
- To overcome those last difficulties, approximating $\mathcal{H}$ by a scalar ω can be proposed. It can be either constant – depending of the problem treated – or chosen at each iteration $\omega^{(k)}$ by methods such as Aitken relaxation or steepest descent (gradient method). In this latter case, the convergence rate (decreasing of the residual norm) is proved to be able to reach $\alpha \simeq 1.6$. The relaxation techniques are detailed in Sec. 2.3.2.

TANGENT METHODS require less iterations than secant methods but imply building tangent matrices and solve linear systems of equations.

- Newton-based algorithms consider the use of the Jacobian of the residual equation (2.10): $\mathcal{J}(\boldsymbol{u}^{(k)}) = \partial_{\boldsymbol{u}}\boldsymbol{r}^{(k)}$ The search direction is then given by $\mathcal{H} = \left[\mathcal{J}(\boldsymbol{u}^{(k)})\right]^{-1}$ This approach exhibits a monotonically decreasing residual norm of order $\alpha = 2$. Unfortunately the computational cost of building this tangent matrix can be very high, and, more problematic, some of the cross terms between fluid and structure subproblems are often not reachable in codes.
- quasi-Newton based algorithms are based on an approximate of the Jacobian and especially the cross terms. Newton and quasi-Newton methods for fluid-structure interaction are described in Sec. 2.3.3.

REMARK: The stability domain of fluid-structure interaction problems is mainly governed by the ratio between fluid and solid density (see Appendix A). For Civil Engineering based problems where we consider water and air against construction materials, this is not a major issue. However, when coupling incompressible flows with structure, the stability region remains limited. A way to extend it is to consider relaxation techniques [Joosten et al., 2009].

2.3.2 *Relaxation techniques*

Relaxation of the interface displacements can be considered as the equivalent to the line-search step of a nonlinear solver [Golub and van Loan, 1996]. Accordingly, all the known line-search techniques can be applied here. The classic update process of the interface displacements is improved by the introduction of a simple scalar, the relaxation, noted ω:

$$\boldsymbol{u}_{N+1}^{(k+1)} = \boldsymbol{u}_{N+1}^{(k)} + \omega\, \boldsymbol{r}\left(\boldsymbol{u}_{N+1}^{(k)}\right) \tag{2.35}$$

A good relaxation parameter is:

$$\left\|\boldsymbol{r}\left(\boldsymbol{u}_{N+1}^{(k+1)}\right)\right\| \leq \left\|\boldsymbol{r}\left(\mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right)\right)\right\| \tag{2.36}$$

*i.e.*the residual of the improved interface displacements that than the residual of the displacement obtained without relaxation.

2.3.2.1 *Fixed relaxation*

The simplest idea is to apply a fixed value to the relaxation parameter ω. A balance has to be found between the large relaxation parameters that are faster but can lead to divergence of the suite of residuals, and small relaxation parameters that allow to control convergence but trigger more iterations.

There is an optimum but it is not known *a priori*. Furthermore, the optimum relaxation parameter is problem-dependent. Last but not least, for nonlinear problems, the optimum value is different at each time step and iteration [Joosten et al., 2009].

REMARK: However, this fixed relaxation technique remains widely used in the domain of Computational Fluid Dynamics. The classic semi-implicit algorithm SIMPLE [Patankar, 1980], used in steady or in large time-step transient computations, in order to enforce the incompressibility condition of discretized Navier-Stokes equations (see Section 1.2.2) converges for a certain range of fixed relaxation parameters [Ferziger and Perić, 1996, Jasak, 1996].

2.3.2.2 *Aitken's relaxation*

Principle of Aitken's relaxation is to use the last two values of residual to improve of the current solution. Consider, for the sake of education, that the interface displacement u and residual r are scalars. The suite of Picard's iterations defined by:

$$u_{N+1}^{(k+1)} = \mathcal{G}\left(u_{N+1}^{(k)}\right) \tag{2.37}$$

is supposed to converge to a solution, as the $\mathcal{G}$ function has a fixed-point. This suite is then improved by simply using the secant method.

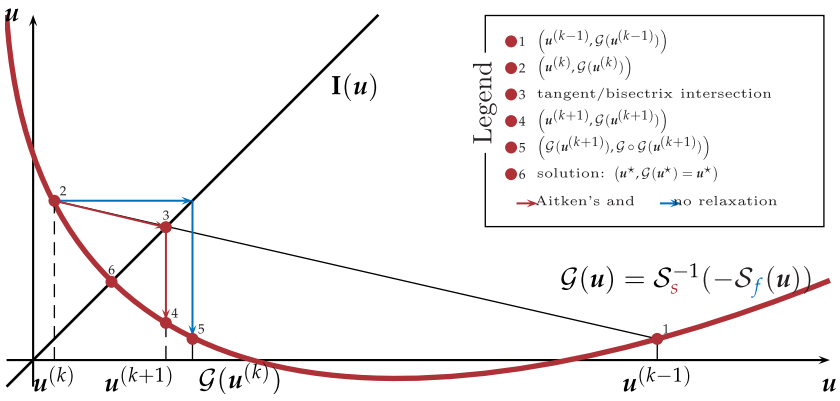


Figure 2.4: Building an improved solution by a secant method: geometric illustration of Aitken's relaxation.

The improved point is defined as the intersection of the secant between points $\left(u_{N+1}^{(k-1)}, \mathcal{G}\left(u_{N+1}^{(k-1)}\right)\right)$ and $\left(u_{N+1}^{(k)}, \mathcal{G}\left(u_{N+1}^{(k)}\right)\right)$ and the bisectrix $y = x$. In Fig. 2.4 a geometrical construction of the improved solution at the intersection of the secant and identity functions is given. Mathematically speaking, it yields the following identity:

$$u_{N+1}^{(k+1)} = \mathcal{G}\left(u_{N+1}^{(k)}\right) + \frac{\mathcal{G}\left(u_{N+1}^{(k)}\right) - \mathcal{G}\left(u_{N+1}^{(k-1)}\right)}{u_{N+1}^{(k)} - u_{N+1}^{(k-1)}}\left(u_{N+1}^{(k+1)} - u_{N+1}^{(k)}\right) \tag{2.38}$$

The intersection equation above gives the improved value of the interface displacements with:

$$u_{N+1}^{(k+1)} = \frac{u_{N+1}^{(k)}\mathcal{G}\left(u_{N+1}^{(k-1)}\right) - u_{N+1}^{(k-1)}\mathcal{G}\left(u_{N+1}^{(k)}\right)}{\mathcal{G}\left(u_{N+1}^{(k-1)}\right) - u_{N+1}^{(k-1)} - \mathcal{G}\left(u_{N+1}^{(k)}\right) + u_{N+1}^{(k)}} \tag{2.39}$$

Looking for an expression of the improved displacements of the form:

$$u_{N+1}^{(k+1)} = u_{N+1}^{(k)} + \omega^{(k)} \left(\mathcal{G}\left(u_{N+1}^{(k)}\right) - u_{N+1}^{(k)} \right) \tag{2.40}$$

gives the following value for Aitken's relaxation parameter $\omega^{(k)}$ from Eq. (2.39):

$$\omega^{(k)} = \frac{u_{N+1}^{(k)} - u_{N+1}^{(k-1)}}{\mathcal{G}\left(u_{N+1}^{(k-1)}\right) - u_{N+1}^{(k-1)} - \mathcal{G}\left(u_{N+1}^{(k)}\right) + u_{N+1}^{(k)}} \tag{2.41}$$

Supposing that the point $u_{N+1}^{(k)}$ was also built by Aitken's relaxation, the following recurrence formula evolves into:

$$\omega^{(k)} = \omega^{(k-1)} \frac{\mathcal{G}\left(u_{N+1}^{(k-1)}\right) - u_{N+1}^{(k-1)}}{\mathcal{G}\left(u_{N+1}^{(k-1)}\right) - u_{N+1}^{(k-1)} - \mathcal{G}\left(u_{N+1}^{(k)}\right) + u_{N+1}^{(k)}} \tag{2.42}$$

From the definition of the residual in Eq. (2.33), it is quite straightforward to compute the relaxation parameter from the last two values of the residual. The only difficulty is to build an equivalent formula for the vector suite of interface displacements $\left(\boldsymbol{u}_{N+1}^{(k)}\right)$. In [Irons and Tuck, 1969], it is suggested to use the inverse vector:

$$\frac{\boldsymbol{r}_{N+1}^{(k)} - \boldsymbol{r}_{N+1}^{(k-1)}}{\left\| \boldsymbol{r}_{N+1}^{(k)} - \boldsymbol{r}_{N+1}^{(k-1)} \right\|^2} \tag{2.43}$$

which is equivalent to projecting in the direction $\boldsymbol{r}_{N+1}^{(k)} - \boldsymbol{r}_{N+1}^{(k-1)}$ and to doing the inverse with the scalar projected value.

From the scalar definition of Aitken's relaxation parameter Eq. (2.42), and the definition of the inverse vector given in Eq. (2.43) we can build the relaxation parameter for the interface displacements as:

$$\omega^{(k)} = -\omega^{(k-1)} \frac{\boldsymbol{r}_{N+1}^{(k-1)} \cdot \left(\boldsymbol{r}_{N+1}^{(k)} - \boldsymbol{r}_{N+1}^{(k-1)} \right)}{\left\| \boldsymbol{r}_{N+1}^{(k)} - \boldsymbol{r}_{N+1}^{(k-1)} \right\|^2} \tag{2.44}$$

The Aitken's relaxation parameter is exact for linear scalar cases. For vector cases, the optimum efficiency of such relaxation parameter is not proved. However, using Aitken's relaxation has shown tremendous performances in terms of convergence rate and computation stabilization for a small computational cost and its easy implementation. Strong arguments can be found in favor of the use of the dynamic Aitken relaxation for fluid-structure interaction in the [Wall, 1999] and associated works [Küttler and Wall, 2008, Mok, 2001].

In [Deparis et al., 2006], slightly different definition of the Aitken relaxation parameter is given, using a vector expression close to Eq. (2.41). The latter is preferred in Eq. (2.44) as only previous residual and relaxation parameter values are used, that seems a little more consistant than mixing them with direct displacements interface. However, the performance seems to be close. Note that in [Küttler and Wall, 2008], authors highlight that the recurrent terms

in Eq. (2.44) are sometimes missing. That naturally leads to a less efficient relaxation parameter.

The last problem to solve with Aitken's relaxation parameter is the guess of the initial relaxation at the beginning of the iterative process for each time-window computation. As can be seen by Eq. (2.44), $\omega^{(k)}$ is built with a division by a difference of residual $\boldsymbol{r}_{N+1}^{(k)}$ and $\boldsymbol{r}_{N+1}^{(k-1)}$. When the convergence is reached at a certain tolerance, the residual can be considered as zero and the division will trigger numerical problems. To avoid this, it is proposed to use as an initial guess $\omega_N^{(0)} = \omega_0 = 0.1$ [Wall, 1999]. We propose here to use $\omega_{N+1}^{(0)} = \max(\omega_N^{(k_{\max})}, \omega_0)$. This initial guess seems to have better convergence properties, especially for the first iterations.

2.3.2.3 *Steepest descent technique*

Contrary to Aitken's relaxation, the steepest descent technique aims at building an optimum relaxation parameter $\omega^{(k)}$ that will minimize a certain cost function in the direction $\boldsymbol{r}_{N+1}^{(k)}$. For a certain merit function ϕ, it follows:

$$\omega^{(k)} = \arg\min_{\omega^{(k)}} \phi\left(\boldsymbol{u}_{N+1}^{(k)} + \omega\, \boldsymbol{r}_{N+1}^{(k)}\right) \tag{2.45}$$

Assuming a smooth merit function ϕ and small residual $\boldsymbol{r}_{N+1}^{(k)}$, it is possible to build an optimum relaxation parameter $\omega^{(k)}$ using a Taylor expansion at the second order:

$$\omega^{(k)} = \frac{\partial_{\boldsymbol{u}}\phi \cdot \boldsymbol{r}_{N+1}^{(k)}}{\boldsymbol{r}_{N+1}^{(k)} \cdot \left[\partial_{\boldsymbol{u}}^2\phi\right] \boldsymbol{r}_{N+1}^{(k)}} \tag{2.46}$$

At this point, the merit function has to be determined. It is often proposed to choose a merit function so that:

$$\partial_{\boldsymbol{u}}\phi\left(\boldsymbol{u}_{N+1}^{(k)}\right) = \boldsymbol{r}_{N+1}^{(k)} \tag{2.47}$$

This assumption only holds if the residual $\boldsymbol{r}$ is the gradient of an unknown scalar function. It leads to a symmetric Jacobian:

$$\partial_{\boldsymbol{u}}^2\phi\left(\boldsymbol{u}_{N+1}^{(k)}\right) = \partial_{\boldsymbol{u}}\boldsymbol{r}\left(\boldsymbol{u}_{N+1}^{(k)}\right) = \mathcal{J}_{N+1}^{(k)} \tag{2.48}$$

However, as noted in [Küttler and Wall, 2008] the convection phenomenon taking place in the fluid part of the coupled fluid-structure interaction problem invalidat the last assumption. But an optimum relaxation parameter associated can be built as such:

$$\omega^{(k)} = \frac{\boldsymbol{r}_{N+1}^{(k)} \cdot \boldsymbol{r}_{N+1}^{(k)}}{\boldsymbol{r}_{N+1}^{(k)} \cdot \mathcal{J}_{(N+1)}^{(k)} \boldsymbol{r}_{N+1}^{(k)}} \tag{2.49}$$

It shows good performance in terms of number of iterations.

The added difficulty compared to the Aitken's relaxation presented above is the need to compute the vector matrix products $\mathcal{J}_{N+1}^{(k)} \boldsymbol{r}_{N+1}^{(k)}$. To do so, two calculations can be proposed:

FINITE DIFFERENCE where the derivative in the direction $\boldsymbol{r}$ is evaluated through the use a small numerical numerical parameter.

APPROXIMATED FLUID DERIVATIVES where some part of the fluid derivative are neglected in order to build an approximation of the Jacobian.

This two numerical methods can be used in a more general context to build an approximation of the Jacobian for all quasi-Newton techniques applied to fluid-structure interaction presented in Section 2.3.3. This is the reason why the steepest descent method does not really make sense in the fluid-structure interaction context. There are far better options than building a scalar approximation of the search direction if one is able to produce an approximation of the Jacobian!

2.3.2.4 *Relaxation seen as an approximative Newton algorithm*

Relaxation techniques are usually only seen as a way to improve the convergence of a fixed-point iteration solver, disconnected from solver techniques that rely on the evaluation of the Jacobian. The steepest descent method have no actual practical interest for the fluid-structure interaction. However it makes clear the link between the relaxation parameter and the inverse of the Jacobian.

The preeminence is given to the fact that relaxation gives a scalar approximation of the inverse of the Jacobian interface. Aitken's relaxation can be considered as a secant method, and Steepest descent method builds the best scalar approximation of this Jacobian.

The BGS algorithm with relaxation from Eq. (2.9) and Eq. (2.35) can be re-written as follows:

$$-\frac{1}{\omega^{(k)}}\left(\boldsymbol{u}_{N+1}^{(k+1)}-\boldsymbol{u}_{N+1}^{(k)}\right)=-\left({\mathcal{S}_s}^{-1}\left(-\mathcal{S}_f\left(\boldsymbol{u}_{N+1}^{(k)}\right)\right)-\boldsymbol{u}_{N+1}^{(k+1)}\right) \tag{2.50a}$$

$$\left[\tilde{\mathcal{J}}_{N+1}^{(k)}\right]\,\Delta u^{(k)}=-\boldsymbol{r}_{N+1}^{(k)} \tag{2.50b}$$

where the approximation of the Jacobian $\tilde{\mathcal{J}}=-\omega^{-1}\mathbf{I}$ is clearly stated.

It is clear that if one builds a scalar approximation of the Jacobian, the computational cost should be small as the search direction is not the good one. So, the fault of this approximation is that it requires more iteration than Newton or quasi-Newton methods.

In the next section we discuss the methods that propose to build either a better approximation or exact Jacobian inverse in order to improve the rate of convergence of the iterative process.

2.3.3 *Newton and quasi-Newton based strategy*

2.3.3.1 *Formulation as a root equation*

As noted in the previous section, algorithms based on fixed point strategy are known to have a slow convergence rate. As shown in Fig. 2.3, this is mainly due to the lack of search direction when the fixed point Eq. (2.9) is being solved. A good way to overcome this difficulty is to give as search direction the tangent of

the nonlinear equation solved, and in this framework, we rewrite this equation as a root equation:

Given: state variable at T_N at the fluid-structure interface Γ and on the fluid Ω_f and structure domain Ω_s
Find: $\boldsymbol{u}_{N+1}$ on $\Gamma \times [T_N, T_{N+1}]$,
Such as:

$$\boldsymbol{r}\left(\boldsymbol{u}_{N+1}\right) = {\mathcal{S}_s}^{-1}\left(-\mathcal{S}_f\left(\boldsymbol{u}_{N+1}\right)\right) - \boldsymbol{u}_{N+1} = \boldsymbol{0} \tag{2.51}$$

This equation can be solved with a classic Newton scheme that is known to be quadratically convergent in the neighborhood of the solution:

$$\mathcal{J}^{(k)} \Delta \boldsymbol{u}_{N+1}^{(k)} = -\boldsymbol{r}\left(\boldsymbol{u}_{N+1}^{(k)}\right) \tag{2.52}$$

where the Jacobian $\mathcal{J}$ is computed as:

$$\mathcal{J} = \partial_{\boldsymbol{u}} \boldsymbol{r} = \partial_{\boldsymbol{u}}\left(\mathcal{S}_s^{-1} \circ (-\mathcal{S}_f)\right) - \mathbf{I} \tag{2.53}$$

Thus, the Jacobian has to be computed at each nonlinear iteration. To be a little more precise, Sec. 2.1.3 has established that the Steklov-Poincaré operators can be depicted as compositions between mesh solver $\mathcal{R}_m$, fluid solver $\mathcal{R}_f$ and solid solver $\mathcal{R}_s$ and restriction operators for the primal $\mathcal{T}_{\boldsymbol{u}}$ and dual $\mathcal{T}_{\boldsymbol{\lambda}}$ quantities at the interface:

$$\mathcal{S}_s^{-1} \circ (-\mathcal{S}) = \mathcal{T}_{\boldsymbol{u}} \circ \mathcal{R}_s \circ \mathcal{T}_{\boldsymbol{\lambda}} \circ \mathcal{R}_f \circ \mathcal{R}_m \tag{2.54}$$

The derivative of these compositions of different operators has to be built, or at least evaluated in some perturbation direction when a GMRES solver is used [Heil, 2004]. In **Algorithm 8** the Schur-Newton-Krylov algorithm is given:

The solving in **Algorithm 8** can be carried out by an iterative free-matrix method such as GMRES. In this case the Jacobian operator $\mathcal{J} = [\partial_{\boldsymbol{u}} \boldsymbol{r}]$ only needs to be evaluated several times against the perturbation of the primal at the interface state perturbation $\boldsymbol{p}$. The GMRES algorithm for fluid-structure interaction is given in the next section.

Note also that the evaluation of the residual amounts to perform an iteration of the DFMT-BGS algorithm and therefore has the same computational cost.

2.3.3.2 *Generalized Minimum Residual Algorithm*

Solving the tangent problem by a GMRES solver requires several evaluations of the Jacobian operator against the perturbation of the primal at the interface $\boldsymbol{p}$. A way to perform this task is to follow the **Algorithm 9** given above.

This Schur-Newton-Krylov strategy is more robust than the block-Gauß-Seidel one. This is due to the use of a Newton solver and it appears that no relaxation is required. However, the main difficulty of the proposed algorithm is to compute the fluid tangent subproblem as it requires the evaluation of the cross derivative of the fluid problem in its moving domain. This task can be performed with the following strategies:

Algorithm 8 Schur-Newton-Krylov algorithm for fluid-structure interaction

1: Given: initial counter $N = 0$, initial time $T = T_0$, final time $T_{\max}$, window size Δt, initial interface displacement $\boldsymbol{u}_0$.
2: **while** $T < T_{\max}$ **do**
3: $(k) = 0$
4: Predict displacement: $\boldsymbol{u}_{N+1}^{(0)} = \mathcal{P}(\boldsymbol{u}_N^{(k_{\max})}, \dot{\boldsymbol{u}}_N^{(k_{\max})}, \boldsymbol{u}_{N-1}^{(k_{\max})}, \dots)$
5: **repeat**
6: Perform a Picard iteration: $\mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right)$
7: Compute residual: $\boldsymbol{r}_{N+1}^{(k)} = \mathcal{G}\left(\boldsymbol{u}_{N+1}^{(k)}\right) - \boldsymbol{u}_{N+1}^{(k)}$
8: Solve: $\left[\partial_{\boldsymbol{u}} \boldsymbol{r}_{N+1}^{(k)}\right] \Delta \boldsymbol{u}_{N+1}^{(k)} = -\boldsymbol{r}\left(\boldsymbol{u}_{N+1}^{(k)}\right)$
9: Update interface primal variable: $\boldsymbol{u}_{N+1}^{(k+1)} = \boldsymbol{u}_{N+1}^{(k)} + \Delta \boldsymbol{u}_{N+1}^{(k)}$
10: **do** $(k) \leftarrow (k) + 1$
11: **until** $\|\boldsymbol{r}_N^{(k-1)}\| \geq \texttt{TOL}$
12: $N \leftarrow N + 1$ and $T \leftarrow T + \Delta t$
13: **end while**

Algorithm 9 Matrix vector product of the Jacobian with a Krylov vector

1: Perform the fluid tangent subproblem:

$$\left[\partial_{\boldsymbol{u}_f} \mathcal{T}_{\boldsymbol{\lambda}} \circ \mathcal{R}_f \circ \mathcal{R}_m\right] \delta \tilde{\boldsymbol{p}} = -\left[\partial_{\boldsymbol{u}_s} \mathcal{T}_{\boldsymbol{\lambda}} \circ \mathcal{R}_f \circ \mathcal{R}_m\right] \boldsymbol{p}$$

2: Perform the solid tangent subproblem:

$$\left[\partial_{\boldsymbol{u}_s} \mathcal{T}_{\boldsymbol{u}} \circ \mathcal{R}_s\right] \delta \boldsymbol{p} = -\left[\partial_{\boldsymbol{u}_f} \mathcal{T}_{\boldsymbol{u}} \circ \mathcal{R}_s\right] \delta \tilde{\boldsymbol{p}}$$

3: Evaluate the Jacobian as:

$$\mathcal{J}\boldsymbol{p} = \left[\partial_{\boldsymbol{u}} \boldsymbol{r}\right] \boldsymbol{p} = \delta \boldsymbol{p} - \boldsymbol{p}$$

FINITE DIFFERENCE APPROXIMATION: The cross-Jacobians of the fluid tangent operator are built with a Finite Difference approximation in the direction of interest [Matthies and Steindorf, 2003, Matthies et al., 2006], and therefore, only the evaluation of state operators is required. However, there is no *a priori* rule to select an infinitesimal finite difference step. Thus, non-consistent Jacobian can be build, which leads to decrease the overall performances of the algorithm.

EXACT NEWTON EVALUATION: In [Fernández and Moubachir, 2005] as well as in [Dettmer and Perić, 2008], the cross-Jacobians of the fluid and solid operators are computed exactly using shape derivative calculus in order to differentiate the integral of the weak state operators with respect to their supports. This method requires modification of the solvers associated with the fluid and solid subproblems. Furthermore, it was developed for coupling problems relying on a weak-formulation, and can not be applied to FV for fluids without developing the corresponding associated theory.

QUASI-NEWTON EVALUATION: Some terms of the shape derivatives are neglected in order to build the Jacobian matrix and inverse it more easily [Dettmer and Perić, 2008, Gerbeau and Vidrascu, 2003b]. This method is also used in the monolithic approach to reduce the cost of building the tangent matrix for the whole coupling problem. It is also possible to build an approximation of the Jacobian using an associated simplified problem whose evaluation gives an accurate approximation of the fluid on moving domain tangent operators [Fernández et al., 2007].

Even if SNK is supposed to converge in less iterations than the block-Gauß-Seidel solver, the cost of computing the Jacobian computing often counters this advantage, and it is *a priori* not possible to say which of the two solvers is least CPU demanding. In [Barcelos et al., 2006], SNK is shown to be faster than BGS when the structure becomes softer, and in [Deparis et al., 2006] the advantage is made obvious when applied to problems coupled with small time steps. However, in [Küttler and Wall, 2008], the maximum gain observed in CPU time for the DFMT-SNK algorithm is around 33% of the CPU time observed for DFMT-BGS.

CLOSURE

This second chapter was dedicated to the introduction of different possible strategies for coupled fluid-structure interaction problems. The monolithic approach is briefly recalled thanks to main references, but the partitioned approach is here preferred for its modularity and the possibility to re-use existing software (eee Chapter 3). The partitioned approach used here is based on the Direct Force-Motion Transfer. Explicit and implicit coupling algorithms for multiphysics problems are detailed.

In the previous Chapter 1, the ALE formulation of the fluid flows as well as the Lagrangian formulation of the structure part are given. The ALE formulation of the Navier-Stokes equations detailed previously apply to incompressible flow in a moving domain. For this range of application, coupling with an explicit

strategy leads to the so-called "Added Mass effect", and for that justifies the use of more costly implicit solvers.

Implicit solvers can use either a fixed-point strategy that is known to be rather slow to converge, or a Newton strategy that requires building up and evaluating the costly Jacobian. In this work, the fixed-point strategy with an adaptive relaxation parameter shows sufficient performances for the aimed computations (see Chapter 4). The stability of the implicit coupling DFMT-BGS algorithms used here is detailed in Appendix A.

3 Building components for fluid-structure interaction

In this chapter we discuss how to build the software tool for fluid-structure interaction problem by re-using existing software for solid and fluid mechanics in a component technology framework. The solid component calls compiled Fortran subroutines of `FEAP` *computer code. The fluid component is based on direct linking with an existing* `C++` *library* `OpenFOAM`*, and allows parallelization through some CTL features. The coupling between components is ensured by a master-code approach, which allows for the second layer parallelization.*

Contents

3.1 Component technology framework 64
3.1.1 Component Oriented Programming paradigm 64
3.1.2 Component-based implementation and its specifics . 65
3.1.3 The middleware CTL 65
3.2 Structure component based on FEAP 66
3.2.1 Interface definition of the structure component 67
3.2.2 Implementation of coFeap 68
3.2.3 Calling a service from the mechanical component . . 70
3.3 Fluid component based on OpenFOAM 71
3.3.1 Component interface and its implementation 71
3.3.1.1 Interface defintion of the fluid component . 71
3.3.1.2 Implementation of ofoam-2 73
3.3.2 RPC versus system calls and file reading performances 75
3.3.2.1 Two implementations of the fluid component 75
3.3.2.2 Performance Comparison between ofoam and ofoam-2 75
3.3.3 Parallel CFD Component Features 77
3.3.3.1 Parallel CFD Component Implementation . 79
3.3.3.2 Parallel CFD Component Performances . . 81
3.3.3.3 Parallelization on the same multi-processor machine 81
3.3.3.4 Parallelization on a cluster 84
3.4 The master code cops 87
3.4.1 Software coupling applied to fluid-structure interaction 87
3.4.2 Component architecture of **co**ps 88
3.4.3 Field interpolation between solvers 88
Closure 91

Scientific computing software for a particular domain is often programmed by domain experts themselves. Indeed, a deep understanding of both physical phenomena and of numerical analysis is required to produce a successful piece of scientific software. It implies that in a certain number of research centers, and far more than in any domains where software are used, only one person takes up the roles of domain expert, software designer, programmer, tester and final-user.

Unfortunately, there are few programmers of scientific computing programs who receive a formal eduction in software engineering – and I was not one of them. It implies that when they want to validate new algorithms and numerical methods or to understand physical phenomena, they often choose to start from scratch. As attractive as this idea may be, it often requires tremendous effort of development, and therefore, scientists remains stuck with low d-o-f problems.

The problem is even a little more different in the domain of scientific computing in engineering, where the ultimate goal of scientists is to propose method adapted to software eventually programmed and used by engineers. For this category of problem, discretization on large meshes with many d-o-f are required [Feyel et al., 1997] – *e.g.* prizes given to [Adams et al., 2004]'s first work reaching the half-billion d-o-f in Finite Element Method. Another illustration of this phenomenon is the role taken by large CFD computations in the development and use of the most powerful computers for meteorological and climatic [Fosdick et al., 1996] as well as oceanic computations [Guyon et al., 1999].

In order to facilitate the development of algorithms, general computational type of software such as Matlab or Octave can be used. If they are relatively efficient for prototyping, it is known that their generality leads to relatively poor efficiency for the domain of our interest.

Another important point to underline is the importance of reliability in the development of new software. As no known human programmer is able to develop a bug-free program, testing and validating a piece of code is often the most time consuming task in the development of scientific computing programs. Furthermore, as the developer is often the tester, it is really rare that systematic proof of the reliability of the program is made. The only solution for tracking bugs is therefore to use and re-use software in different contexts.

Hence, the re-use of existing tools – software or library – is a major trend in computing. In this work, the component-oriented framework that allows to re-use existing codes for the fluid-structure interaction paradigm that was mathematically set in the previous Chapters is described. The first steps are to develop a Structure and a Fluid components of their respective subproblems.

For the structure component, the idea is to re-use the existing Fortran-based FEM code `FEAP`. Its implementation and particularly the couping between the Fortran part and the `C++` part of the component is given in Sec. 3.2. The fluid component ofoam is based on the `C++` programmed libraries of OpenFOAM. Its implementation as well as performance comparison are detailled in Sec. 3.3. The performances of the Remote Process Execution is compared to file reading/writing and program execution. The tremendous cost of some fluid computations justify the development of an ofoam version that re-use the parallel features of the existing program in the way components would use it. The coupling

of the components is ensured by a master-slave approach, developing a `C++` component that implements the coupling algorithms defined in Chapter 2. The interpolation of fields between the two solvers is insured by a specific component developed by [Jürgens, 2009].

3.1 Scientific computing design in the component Technology Framework

3.1.1 *Component Oriented Programming paradigm*

As described below, the need to re-use existing software products and tools in a more general context is a major trend of software engineering. In the famous Garmish (Germany) NATO conference that took place in 1968, the first stones of modern software engineering where laid, and some answers to the problems described previously, and more generally, to the so-called Software Crisis were given [O'Regan, 2008]. In [Mac Ilroy, 1968] for instance, the notion of components and their role in software re-use was introduced for the first time, but a proper definition is yet to be accepted by everybody. For instance, a loose definition of components as entities which can be used, possibly with modifications, in a new software development can be found in [Coulange, 1998, Berti, 2002]. With a more specific meaning, in [Szyperski, 1998] ones defines a component as a piece of software capable of performing certain tasks prescribed within its interface and which it is able to communicate with other components. In the present case, latter definition is considered as the more stringent .

This paradigm somehow extends the concept of object in object-oriented programming with the notion of communication between components. The generalization of the definition of class methods – for the oriented-object programming – equals here an interface listing the tasks it can execute on input data or on its attribute. This interface can be seen as the contract that links the two parties: components and clients. Another important feature of components is that they are deployed on a system. Or as stated in [Szyperski, 1998]:

> Software components are binary units of independent production, acquisition, and deployment that interact to form a functioning system

Thus, in order to be considered as a component, binaries should obey the five following criteria as specified in [Niekamp, 2005b]:

Multi-usability: several components should be instantiated on the same CPU or on different machines; this feature allows parallel processing.

Non-context specific implementation: two components that fit the same interface can be used by a service without any modification. For example, in this work, the structure component coFeap based on the FEM code `FEAP` [Taylor, 2008], can also be replaced by the component based on `ParaFEP` software product [Niekamp and Stein, 2002].

Composability: it is possible to make components out of components. We propose to couple the fluid and structure components with a master com-

ponent **co**ps (COupling COmponents by a Partitioned Strategy). A version of this component is used to perform stochastic computations on fluid-structure interaction problems [Austruy, 2008, Austruy et al., 2008].

ENCAPSULATION: it is not possible to access the inner structure or details of a component via an interface. Hence, the implementation of the algorithm defined by the interface and the implementation of the communication method are strictly separated.

DEVELOPMENT AND VERSION INDEPENDENT: a component defined through its interface should be independent from its version, programming language and even of the compiler used. It means that once deployed on a machine, it can be called by any service knowing only its interface and the middleware until it is erased.

More details can be found in the following conversation [Broy et al., 1998].

3.1.2 *Component-based implementation and its specifics*

From a scientific computing point of view, the implementation based on component technology relies on the definitions of API (Application Program Interface). Indeed, both clients – those that call a method – and services – those that execute the task – have to share the knowledge of classes, functions and methods available. The API allows the Remote Procedure Call (RPC), or yet called Remote Method Invocation (RMI), to access libraries, procedures or stored object throughout a network or on the same machine.

In a component-based implementation, a strict separation is made between the algorithm and the method implementation on one side, and the communication handled by the middleware on the other side. Ideally, the client side is totally independent from the way the clients are called, and it does not matter whether they are available locally or remotely called.

In scientific computing, RPC are rivaled by explicit Message Passing API. The main drawback of Explicit Message based programs is that they mix what concerns the algorithm with what concerns the communication between pieces of software. However, from a historical point of view, they were the first to be used in this domain. Thus, MPI [Gropp et al., 1994] or PVM [Geist, 2007] initially only allowed Explicit Message Passing. Nevertheless, the advantages of RPC is now generally admitted, as confirmed by the growing number of types of middleware products available for scientific computing.

3.1.3 *The middleware CTL*

As depicted in Section 3.1.3, component based development requires a middleware layer between clients and services. More precisely, in [Orenstein, 2000], a middleware is defined as:

> Middleware works by providing a standardized, API-like interface that can allow applications on different platforms or written in different languages to interoperate.

Many free and non-free middleware are currently available on the market; The most well known are, CORBA (Common Object Request Broker Architecture), Java™RMI or Microsoft® .NET. However, the field of scientific computing requires very high performance in communication, which implies that only a few of the available middleware are of interest for extensive computation. In fact, according to information on performance computing between different types of middleware in [Niekamp, 2005b], only CORBA – among the quoted environments – fulfills the cost requirement, but is known for its complicated syntax.

In the last ten years, new components like CCA [Kohl and Bernholdt, 2002], Charm++ [Lawlor and Zheng, 1999] or CTL where specifically developed for the need of scientific computing and with the aim of simplifying the syntax. In this work following in the steps of earlier development [Niekamp et al., 2009], we will use the CTL as middleware.

Initially a part of `ParaFEP` [Niekamp and Stein, 2002], the Component Template Library (CTL) was developed by Dr. R. Niekamp at the Institute für Wissenschaftliches Rechnen (TU-Braunschweig). It is a `C++` template library, like the STL, that builds a wrapper or a communication layer around a software, and thus allows so to build components from existing piece of code. This layer ensures a serialization of the data to be exchanged over a network, and implements, via code generation, an interface defined in a particular header file. Unlike complicated CORBA syntax, the API is here written in C-preprocessor language, often in a `*.ci` (for Component Interface) file.

The two main advantages of CTL are [Bügling, 2006]:

- providing a lightweight that can be used on top of several local (library and thread) or remote communication methods (TCP/IP, MPI and others).

- making the process of writing an application or a service which uses the CTL protocol as transparent as possible. Developers of a service can write its implementation like they would write a normal local class, with the exception that they need to give the CTL a method to serialize the contained data. Developers of a client only needs then to choose a service within the CTL API and how it starts. They can use the objects provided by CTL services as if they were standard local objects.

3.2 Structure component based on `FEAP`

Building a CTL component based on `FEAP` was the first contact of the research group of Adnan Ibrahimbegović at *LMT-Cachan/Civil Engineering* with the component technology. The first prototype was designed by Pr. R. L. Taylor and Dr. R. Niekamp and co-workers [Niekamp et al., 2009] to match a generic interface that every "mechanical simulator" should fulfill (see Section 3.2.1). This first prototype has been fully re-written, extended and maintained by Dr. M. Hautefeuille and myself, and the architecture and its implementation is described in Section 3.2.2. For more details, the reader is however invited to read coFeap's manual [Kassiotis and Hautefeuille, 2008].

3.2.1 *Interface definition of the structure component*

Building a component requires first to define the list of methods that it should perform. The methods are defined by their names and a list of inputs and outputs (I/O). To this method, one (or more) constructors and optionally one destructor are added. This defines the component interface that is written in a file in C-prepocessor language when the CTL is used (other middleware often proposing their own interface definition langage). For CTL component, the interface file is often marked by the extension `.ci` for *Component Interface*.

In a weak sense, defining an interface is really close to declaring a class in all object oriented languages. Thus, the `.ci` file can be seen as equivalent to an `.hpp` file that contains the declaration of a `C++` class [Stroustrup, 1986].

The interface design was first seen as a general abstraction of what defines all simulators in the field of mechanical engineering. Subtle distinction between Fluid and Structure domains lead to the introduction of a simulator interface: the `simu.ci` for mechanical components detailed below, and the `CFDSimu.ci` described in Section 3.3. The `simu.ci` interface is made to match any FEM code. Once the connect process via CTL is done, the relying code behind is used as a black box by the client calling this service. For instance, our coupling component can call for solid mechanics indifferently either coFeap or the CTL component based on `ParaFEP` [Niekamp and Stein, 2002] that matches the same interface.

The `simu.ci` looks like the following: the method and constructor declarations lie between the `CTL_ClassBegin` and the `CTL_ClassEnd`.

```
#ifndef __SIMU_CI_
#define __SIMU_CI_

#include <ctl.h>

#define CTL_ClassTmpl SimuCI, ( scalar1 ), 1
#include CTL_ClassBegin

// Constructors and methods declaration

# include CTL_ClassEnd

#endif//__SIMU_CI_
```

Basically, the methods can be split into three generic distinctions:

PRE-TREATMENT focuses on the definition and set times up of the discretized problem. Defining the geometry and zone of interest, getting the nodes, the elements and their connectivity and applying loadings to the structure. For instance, the `set_load` applies a vector of load to the structure nodes. This loading can be primal (imposed displacement), dual (imposed forces) or both.

```
#define CTL_Method6 void, set_load, (const array<scalar1>/*value*/), 1
```

TREATMENT build residual and solve linear and non-linear problems defined in the component. For instance the method `solve` performs the resolution of a linear or non-linear set of equations to a tolerance previously given to the component for one time step.

```
#define CTL_Method10 int4, solve, (), 0
```

POST-TREATMENT is there to get the desired variables (primal, dual or internal variables for the case of non-linear solid mechanics problem) either in the form of field that can then be manipulated by a client, or saved into files to be plotted with a visualization tool such as Paraview [Kitware Inc., 2009]. The `get_state` allows to get variables of different kind (defining by the character chain `disp`, `velo` or `acce` for instance) in the region of interest.

```
#define CTL_Method25 void, get_state,
    (const string/*name*/, array<scalar1>/*value*/) const, 2
```

Therefore developing a component such as coFeap means giving an implementation to the method declared in the component interface `simu.ci` in a way such as the existing `FEAP`library is called. The next subsection deals with this problem.

3.2.2 *Implementation of coFeap*

The Finite Element Application Program (`FEAP`) has been developed by Pr. R. L. Taylor and his co-workers since late 70's. At that time, `Fortran 77` was the most widely used programming language in the field of computational science. Thus, most of the `FEAP` code was developed in that language, with some more recent parts written in `Fortran 90` for specific issues, and in `C` for all that concerns dynamic memory allocations. Since 7.5 version of `FEAP` supported compilation on 32 or 64 bit machines, and we retain this version to build our component. Therefore, one of the challenges when building coFeap is to connect a `Fortran` based code in the CTL middleware programmed in meta-template `C++`, which has been addressed in [Niekamp et al., 2009]

Despite the language small heterogeneities described above, the inner structure of `FEAP` is rather simple, which made it an interesting code for the development of coupled software product. The left part of Fig. 3.1 describes the way the compilation of `FEAP` is done. The header files contain the global variables used in the subroutines. Most of those are implemented in `Fortran *.f` files. They are compiled and gathered in the archive `Feap7_5.a` which is the only library as well as the core of the code. A main routine, `feap75.f` is then compiled and linked to this archive in order to build up the executable file `FEAP` that can then be called to performs Finite Element computations.

Once the `FEAP`archive is built, the building of the component can be performed following the central part of Fig. 3.1:

METHODS COMPILATION: Each method that is declared in the `simu.ci` file needs to be implemented in a `Fortran` subroutine file. These files bear the prefix `simu_` and the suffix `_impl` in order to allow the connection to the interface. All the routines in the `simu_*_impl.f` files are compiled independently.

INTERFACE CONNECTION: When a component is based on a `C++` class, the connection to the interface is made by binding each method defined within

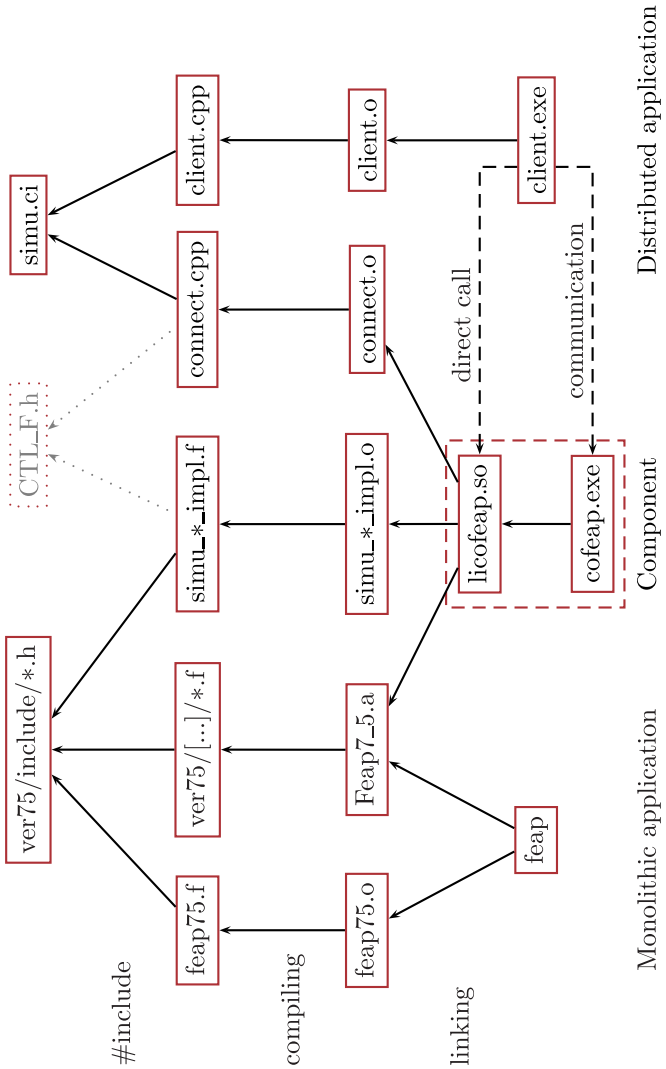


Figure 3.1: coFeap dependencies graph

the interface to a public method of the class. In `Fortran`, as there is no difference between the declaration and the actual implementation of a method, another approach has to be used. The CTL behaves like if a virtual header `CTL_F.h` was created for the `Fortran` subroutines. The `connect.cpp` files take the following form

```
#define CTL_ConnectF
#define CTL_ClassPrefix simu
#include <simu.ci>

void CTL_connect()
{ ctl::connectF<SimuCI<double>, ctl::Extern::SimuCI>(); }
```

The default behavior is handled by the `CTL_connectF` library that proposes to connect all methods whose names begin by the `CTL_ClassPrefix`, here `simu`, as mentioned in the previous paragraph.

LINKAGE TO THE ARCHIVE: The last step is the linkage of the binaries obtained from the method `simu_*_impl.o`, the connection step `connect.o` and the `FEAP`'s archive file `Feap7_5.a`. Two output files are built from this linkage process: first a library `libcofeap.so` that can be used through CTL with dynamic linkage or thread calls; then a remote executable `cofeap.exe` that can be called via TCP through sockets of a network, via pipe to avoid firewall or via MPI or PVM if this protocols are installed.

Once those two steps done, it is possible to call coFeap as a service by any of the direct calls (dynamic linkage, thread...) or communication processes (TCP, pipe...) supported by the CTL. Examples will be shown in the next Section 3.2.3.

3.2.3 *Calling a service from the mechanical component*

The right part of Fig. 3.1 depicts the compilation process of a client that may use the RPC of any `SimuCI` service that fits the `simu.ci` interface, and especially coFeap. In this particular case, the client is a `C++` code, `client.cpp`, which includes `simu.ci`. After compilation and linkage, the resulting executable `client.exe` can directly call the shared library `libcofeap.so`, via dynamic linkage or thread on the same machine, or communicate with the remote executable `cofeap.exe`, via `tcp` call on a remote host.

Thus the initialisation process is carried out as follows:

```
include <simu.ci>
int main () {
    ctl::location loc;
    ctl::link link ( loc );
    SimuCI < double > cofeap( link, "Cfile" );
    return 0;
}
```

The string `"Cfile"` describes the name of the control file. It is in charge of the behavior of the component (deciding for instance, whether the boundary condition that are send to the component correspond to force loading or imposed displacement). For using the shared library, string `loc` has to take the following value:

```
loc = "<path>/libcofeap.so␣-l␣lib";
```

On the contrary, calling a remote process on a machine named `retsina` has to be:

```
loc = "retsina:<path>/cofeap.exe␣-l␣tcp";
```

Once the component is instantiated, the client does not have to know how the CTL handles the communication, and each method of the component is called as if it were a classic class. Thus, it is possible to build the Poincaré–Steklov operator described in **Algorithm 4** using the pre- and post-treatment methods described above.

```
double dt = 0.1;
std::vector<double> l, u;
l = [ ... ];

cofeap.set_load(l);
cofeap.time_step(dt);
cofeap.solve();
cofeap.get_state("disp", u);
```

Another choice could have been to build directly the Poincaré–Steklov operator at the level of the component, defining it in the interface. However, for the sake of modularity, it has been chosen to declare and implement only elementary methods at this level. In our implementation, the Poincaré–Steklov operator is implemented in the form of a template class `PoincareSteklov` that accepts any `simu` component that fits some of the requirement of their interfaces. For more details, see Sec. 3.4.2

3.3 Fluid component based on OpenFOAM

Building a component based on existing CFD solver is not an easy task. First, one has to choose a discretization strategy. As stated in [Ferziger and Perić, 2002], among all the CFD software disposable on the market, many more are FV- rather than FE-based.

Thus, to illustrate the generality of the coupling approach, we choose to couple an FE code for the structure to an FV code for the fluid. As I was not specialist of CFD, another criterion was to choose a software with an active community. Last but not least, developing components is easier when one has access to the source code.

All these reasons make that `OpenFOAM` was first chosen to develop a CFDsimu component.

3.3.1 *Component interface and its implementation*

3.3.1.1 *Interface defintion of the fluid component*

As for the structure component, building the CTL CFD component requires first to define the list of methods that it should perform (Section 3.2.1) in C-preprocessor language.

The `CFDSimu.ci` could have been seen as a specialization of the generic `simu.ci` interface that defines a structure component, and more generally, of any continuum mechanic solver. However, as the development of the `CFDSimu.ci` and `simu.ci` were made in parallel, based on two very different software products, this was not noticed at first glance.

The `simu.ci` looks like the following: the methods declaration and the constructor lie between the `CTL_ClassBegin` and the `CTL_ClassEnd`.

```
#ifndef __CFDSIMU_CI_
#define __CFDSIMU_CI_

#include <ctl.h>

#define CTL_Class CFDSimuCI
#include CTL_ClassBegin
```

```
  #define CTL_Constructor1 ( const string /*controlFile*/ ), 1

  // methods declaration

#include CTL_ClassEnd
#endif
```

Here, the constructor accepts as an argument a string that contains the path to a control file, read by the `boost_program_option` library [Schäling, 2009]. This control file contains information such as the dimension of the problem, the name of the solver invoked, that of the interface and the way to get values when sub-iteration time steps are performed.

Basically, CFD component methods can be split into three generic groups:

PRE-TREATMENT focuses on the definition and set up of the discretized problem. Basically, it concerns the application of boundary conditions on the CFD problem. The boundary conditions can be of Dirichlet (primal, *i.e.* domain displacements or fluid velocity at the interface) or von Neumann (dual, *i.e.* pressure at the interface).

```
#define CTL_Method1 void, set,
    ( const string /*name*/, const array<real8> /*v*/ ), 2
```

TREATMENT methods build residual and solve linear and non-linear problems defined in the component. For instance the method `solve` work out the non-linear transient Navier-Stokes equation on a moving domain on a window of size δt.

```
#define CTL_Method3 int4, solve, ( const real8 /*timeStep*/ ), 1
```

As the time integration on the window can be performed with many small time steps, we also define a `goback` method that allows to goback in time to the beginning of the window, where the state of the computation was previously saved. This method is useful for implicit coupling and convergence difficulty in the given step.

```
#define CTL_Method4 void, goback, (), 0
```

POST-TREATMENT is there to get the desired variables (primal or dual) in the form of an array that can then be manipulated by a client. It allows to get variables of different kind (defined by the character chain `disp`, `velo` or `forc` for instance). The interface of interest is defined in the control file according to the boundary names of the `OpenFOAM` case treated.

```
#define CTL_Method2 void, get,
    ( const string /*name*/, array<real8> /*v*/ ) const, 2
```

We also propose an error estimate that evaluates the discretization error with a residual based technique [Jasak and Tuković, 2007].

```
#define CTL_Method6 void, error, ( real8 /*err*/ ) const, 1
```

Therefore developing a component such as ofoam means implementing to the method declared in the component interface `CFDSimu.ci` in the same way as the existing `OpenFOAM` libraries are called. The next subsection deals with this problem.

3.3.1.2 *Implementation of ofoam-2*

Open Source Field Operation and Manipulation [OpenCFD LTD, 2009] (`OpenFOAM`) is first and foremost a `C++` library. It is used primarly to build executables, known here as *applications*. Applications are mainly of two kinds: *solvers* – designed to solve a specific continuum mechanics problem – and *utilities* – designed to manipulate data, mainly during the pre- and postprocessing phases. The development of `OpenFOAM` was both held by academics (London Imperial College) and a company named OpenCFD LTD. It started at the beginning of the 90's and made this library one of the first entirely developed in `C++` in the field of computational mechanics. The code was made freely available during the 2000's. For more details, the reader is invited to the following publications [Jasak, 1996, Weller et al., 1998] and refered to the active forum[1].

Despite being entirely programmed in `C++` in a clear and consistent style, the extensive use of object-oriented advanced features makes the deep understanding of `OpenFOAM` quite obscure at first. However, the programming of new applications is really facilitated by the clear syntax that allows to write solvers and utilities in a style really close to the continuum mechanics formulation [Weller et al., 1998]. The left part of Fig. 3.2 describes the way the compilation of *solvers* and *utilities* is done. Classes are declared in various `*.H` files, implemented in `*.C` in order to build different libraries `lib*.so` associated to the various ranges of applications (*e.g.*: general FV libraries, moving mesh libraries or fluid model from laminar stationary to turbulent dynamic flow libraries...). Applications for each kind of finite volume solvers desired are implemented in different `*.C` main routine, dynamically linked to the compiled libraries.

From the `OpenFOAM` general overview given above, we decide to build a component following the idea given in the central part of Fig. 3.2.

Method implementation: A template wrapper class, named ofoam, that implements the methods required by `CFDSimu.ci` is written. The methods are implemented in in `*.inl` files.

Interface connection: Contrary to what is explained for coFeap's component based on the `Fortran` software `FEAP`, the CTL does not deal with the connecting process. However, it is obvious that each public method of the ofoam template class corresponds to a method of the `CFDSimu.ci` interface. The `connect` thus takes the following form:

```
#define CTL_Connect

#include <cfdsimu.ci>
#include <ofoam.hpp>
#include "connectDetails.hpp"
```

[1] http://www.cfd-online.com/

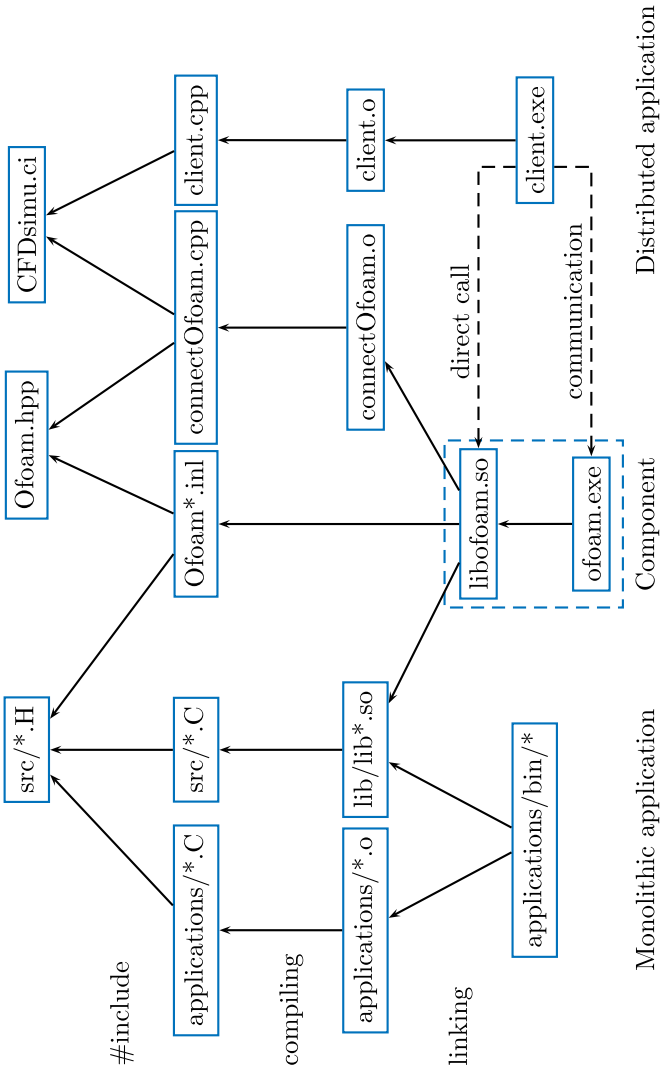


Figure 3.2: ofoam-2 dependencies graph

```
void CTL_connect() {
    ctl::connect<CFDSimuCI, Ofoam<>, connectDetailsCI >();
}
```

where `connectDetailsCI` binds each ofoam public method to its interface declaration.

Linkage to archive: The last step is the linkage of the result of the connection step with the `OpenFOAM` shared objects. Special care has to be taken in handling dynamic libraries, both in `OpenFOAM` and in the CTL component. We here choose to follow what is done with `OpenFOAM`, and to use `dlopen` with `RTLD_LAZY`||`RTLD_GLOBAL` opening options.

Once these two steps are done, it is possible to call ofoam-2 as a service by any of the direct calls (dynamic linkage, thread...) or communication processes (tcp, pipe...) that are supported by the CTL.

3.3.2 *RPC versus system calls and file reading performances*

3.3.2.1 *Two implementations of the fluid component*

The first version of quick implementation was implemented by M. Krosche, in a collaborative project between the Institut für Wissenschafltiches Rechnen (TU-Braunschweig) and BMBF. As explained in [Krosche, 2009], the first ofoam component architecture is based on file reading and system calls. Behind the `CFDSimu.ci` interface is implemented a class that reads output files from `OpenFOAM` and uses system calls in order to perform computation. This has the advantage of being not intrusive at all, and this version of the component depends neither on header files nor on `OpenFOAM` libraries. So, it is allowed to distribute this component in any Licence (here LGPL was used) even if `OpenFOAM` is GPL.

We chose in 2008 to develop the new version of ofoam described above in Section 3.3.1.1. As the distribution and licensing of the component was not a major issue, it was decided to develop a component directly linked to `OpenFOAM`. As seen in Section 3.3.2.2, this has the major advantage of working faster.

To evaluate the performance gain obtained with the new implementation, we propose to compare the CPU time required by the call of methods of each component on the same test case.

3.3.2.2 *Performance Comparison between ofoam and ofoam-2*

The tests are performed on a Newtonian flow in a cylinder for the two architecture of components: ofoam– based on file reading – and ofoam-2 – the wrapper class and component build around `OpenFOAM`.

The dimensions of the tube are a $5m$ diameter and a $10m$ length, and the computation pertains to fully three dimensions flow. The input velocity on the inlet is imposed as being uniform ($10m.s^{-1}$), whereas at the outlet, the pressure gradient is set to 0. Perfect wall conditions (zero velocity) are applied to tube walls. We consider as material properties the following values for the density $\rho = 1.0kg.m^{-3}$ and dynamic viscosity $\mu = 0.01m^2.s^{-1}$. The numerical simulation spans from time $t = 0s$ to time $t = 0.01s$.

The discretization in space is carried out by second order Finite Volume approximation. In time, an implicit Euler scheme with $dt = 0.001s$ is applied. The algorithm chosen is based on PISO [Ferziger and Perić, 2002, Jasak, 1996]. At each time step, two outer corrections are performed to ensure the pressure velocity coupling. The solvers for the fluid velocity and pressure fields are based on PBiCG and PCG respectively.

The study is performed for meshes with 9×10^3, 72×10^3 and 576×10^3 cells (the number of d-o-f equals roughly 4 times the number of cells). In Fig. 3.3, the coarsest mesh is given. The velocity field magnitude for the final time step is represented in Fig. 3.4, indicating that the parabolic profile expected away from the inlet is well represented.

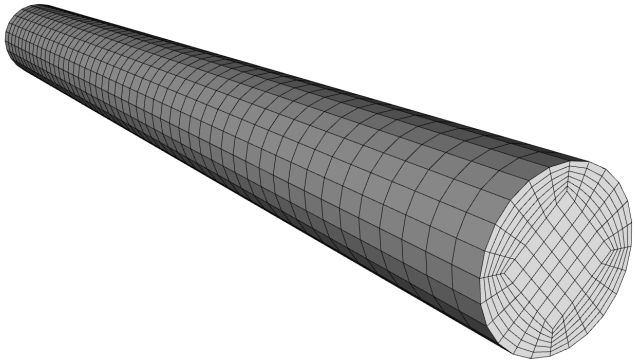


Figure 3.3: Cylinder discretized with 9×10^3 cells

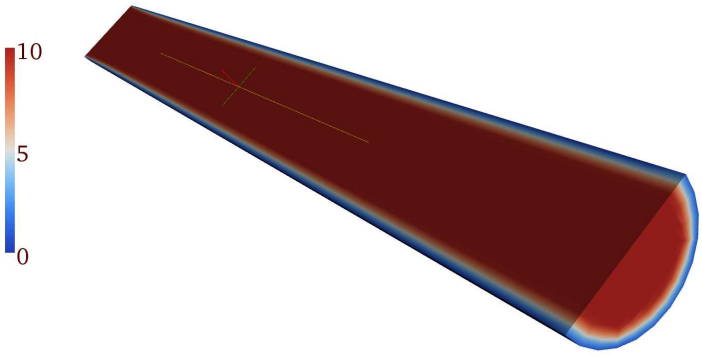


Figure 3.4: Velocity field magnitude in the cylinder in $m.s^{-1}$

The overall performance of components is the one that was expected. The methods can be divided into two classes:

- methods that require intrinsically a lot of data exchange compared to the computation cost – `get` (nodes, pressure, velocity) and `set` (velocity, mesh displacements)
- methods for which most of the time is spent for computations – `solve`, `init`.

For the former, a component based on file reading will be intrinsically less efficient, since the speed of access to data by opening and reading files written on the hard drive cannot compete with a direct access to the memory. For the latter, the performance mainly depends on the implementation of `OpenFOAM`

cells		9×10^3		72×10^3	
version		ofoam-**2**	ofoam-1	ofoam-**2**	ofoam-1
methods	`init`	5.03×10^{-01}	5.50×10^{-01}	$3.96 \times 10^{+00}$	$3.79 \times 10^{+00}$
	`getnodes`	1.44×10^{-04}	1.75×10^{-02}	3.10×10^{-04}	1.34×10^{-01}
	`set`	5.07×10^{-05}	5.27×10^{-04}	8.92×10^{-05}	1.46×10^{-03}
	`solve`	6.10×10^{-01}	9.66×10^{-01}	$1.60 \times 10^{+01}$	$1.73 \times 10^{+01}$
	`get`	2.00×10^{-04}	3.48×10^{-02}	6.29×10^{-04}	2.68×10^{-01}
	total	$1.17 \times 10^{+00}$	$1.89 \times 10^{+00}$	$2.00 \times 10^{+01}$	$2.19 \times 10^{+01}$

cells		576×10^3		4608×10^3	
version		ofoam-**2**	ofoam-1	ofoam-**2**	ofoam-1
methods	`init`	$4.52 \times 10^{+01}$	$4.13 \times 10^{+01}$	$5.79 \times 10^{+02}$	$5.50 \times 10^{+02}$
	`getnodes`	8.81×10^{-04}	$1.10 \times 10^{+00}$	3.18×10^{-03}	$8.41 \times 10^{+00}$
	`set`	2.06×10^{-04}	4.74×10^{-03}	6.33×10^{-04}	1.78×10^{-02}
	`solve`	$1.89 \times 10^{+02}$	$1.98 \times 10^{+02}$	$1.87 \times 10^{+03}$	$1.94 \times 10^{+03}$
	`get`	2.22×10^{-03}	$2.09 \times 10^{+00}$	8.64×10^{-03}	$1.77 \times 10^{+01}$
	total	$2.35 \times 10^{+02}$	$2.44 \times 10^{+02}$	$2.45 \times 10^{+03}$	$2.53 \times 10^{+03}$

Table 3.1: Performance comparison between ofoam and ofoam-2 in terms of CPU time for each method given in seconds; the latter has been measured using the CTL profiler (`CTL_Profile`).

itself, as the time spent in data exchange is very little compared to the one spent in the computation, especially for a huge number of d-o-f.

In Fig. 3.5, the bottleneck of a fluid computation pertains, as expected, to the problem solving (matrix inversion for any method used). Note that the `solve` method is a bit faster for ofoam-2 implementation since there is no need for the software to reload data before performing computation. Thus, the slight advantages offered in terms of CPU time does not alone justify the choice of building a new component ofoam-2. Rather the possibility of reusing any class developed in the `OpenFOAM` project provides a better argument in favor of the new implementation

Note also that the perfomances of the `goback`, which allows to go-back to the beginning of the coupling time window considered, is somewhat faster on the first implementation. Indeed, in ofoam-1 it is allowed to go back in time to values previously saved in files, whereas ofoam-2 needs to double the memory allocation size in order to save the values at the beginning of the time windows.

3.3.3 *Parallel CFD Component Features*

For most fluid-structure interaction problems, especially in the development phase of the coupling software, the bottleneck of models are flow computations. Therefore, even if the parallelization of the coupling algorithm (See for instance Sec. 2.2.5) is of interest, this is only the case if the fluid and the structure computations require the same CPU time. The implementation of parallel coupling algorithms can be then accomplished with CTL in a quite straightforward man-

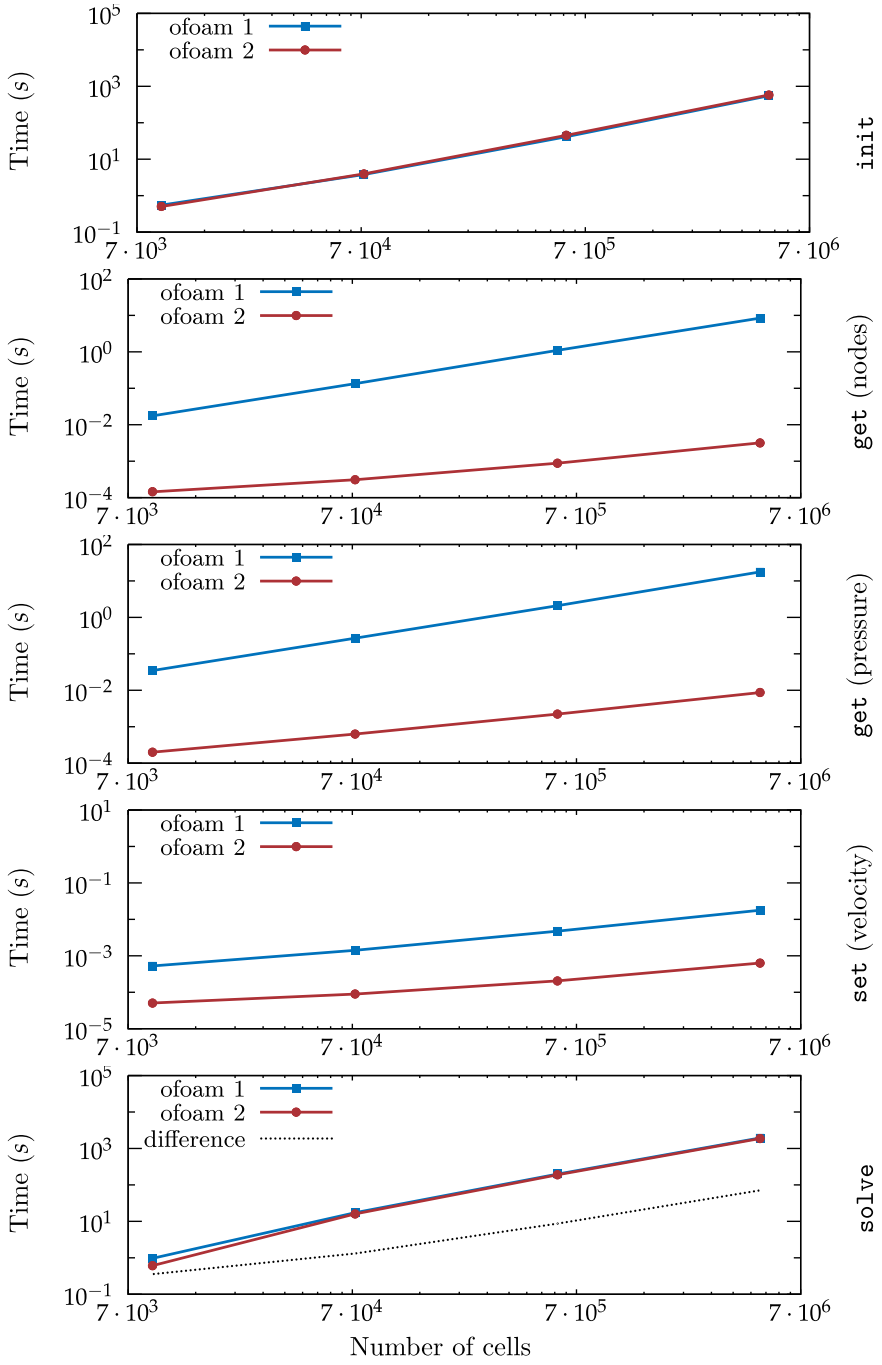


Figure 3.5: Performance comparison between some ofoam and ofoam-2 methods for different mesh refinements.

ner. For instance, in [Hautefeuille, 2009], a parallel implementation of the three field coupling strategy based on LLM is proposed using CTL.

We will only present in detail, the inner parallelism of the fluid component, and how the CTL handles this feature, as explained in the next section.

REMARK: for the computations carried out in this work, it is not required to consider parallelization of the structure subproblem [Farhat and Roux, 1994], but the same considerations can hold when one builds a component based upon a FEM code with parallel features.

3.3.3.1 *Parallel CFD Component Implementation*

Most of current CFD codes, if not all, propose parallel computations feature. It is true that flow simulations is known to be one of the most challenging domains of scientific computations, therefore these problems were among the first to run on clusters (meteorological simulations for weather forecast run every day on some of the most powerful computers in the world to mention only one).

Taking advantage of the inner parallelization of the underlying code is an important requirement when one calls upon a time consuming component. For a client calling a service, it should make no difference to call an ordinary sequential or a parallelized version of the component. It is however challenging to implement such a feature, since the inner parallelization of the existing software often relies on Explicit Message Passing via PVM, MPI or MPCCI. Such implementations thus requires additional level of communication that leads to [Niekamp, 2005b]:

INSERTING OF COMMUNICATION POINTS INTO SOURCE CODE: thus not only the source code but also an expertise for each parallelized program is needed.

NO SEPARATION OF COMMUNICATION AND ALGORITHM: increases the difficultly of code maintenance.

NEW PAIR OF COUPLING: this produces additional amount of programming.

However, as the CTL supports the concepts of processes and process groups with intra- and inter-communication [Niekamp, 2005a], we propose to build a parallel version of the ofoam-2 component in a generic way.

Considering that the implementation of the serial component based on `OpenFOAM` has been performed and matches the `CFDSimu.ci` interface described in Sec. 3.3.1.1. A new wrapper class, the CFDSimuMPI class is built and glued to the same `CFDSimu.ci` interface by another `connect`. This wrapper class does not see directly the wrapper class ofoam-2, but calls components and is therefore independent from implementation.

The first step is to derive from the current CFDSimuCI interface a new Parallel Interface, CFDSimuPI with a specific contructor where parallelization features is indicated by a non-zero value of the switch `bool`:

```
#include <cfdsimu.ci>

#define CTL_Class CFDSimuPI
#define CTL_Extends (CFDSimuCI), 1
#define CTL_Constructor1 (const std::string, const bool), 2
```

```
#include CTL_ClassBegin
#include CTL_ClassEnd
```

This Parallel Interface is then connected to the implementation of ofoam-2 by a `connectDetailsPI` structure that binds the Parallel Interface to the implementation of the wrapper class ofoam-2.

```
#include <cfdsimu.ci>
#include "connectDetails.hpp"

void CTL_connect()
{
    ctl::connect<CFDSimuPI, Ofoam<>, connectDetailsPI >();
}
```

It is now possible to declare and implement the new wrapper class CFDSimuMPI, that only sees the CFDSimuPI interface and to propose a parallel implementation of all the methods required by the Component Interface `CFDSimu.ci`. Hereby is introduced a shortened version of the CFDSimuMPI class, where only the constructor and the solver are implemented (other methods being less demanding, mainly requiring concatenation of results obtained from each `worker`).

```
class CFDSimuMPI{
    ctl::group G;
    ctl::vector<CFDSimuPI> worker;

public:
    // constructor
    CFDSimuMPI(const std::string& cFile){
        G = ctl::getMainGroup();
        worker.resize(G.size());
        ofoamPreTreatment<std::string> preTreat( cFile );
        for(int p=G.size()-1; p >=0 ; p--)
            new (&worker[p]) CFDSimuPI( G[p], cFile, true );
    }

    // method solve implemented in parallel
    int solve( const double timeStep ){
        std::vector< ctl::result<int> > res(G.size());
        for(size_t p=1; p < G.size(); p++) res[p] = worker[p].solve(timeStep);
        int resSolve = worker[0].solve(timeStep);
        return resSolve;
    }

    // other methods
    [...]
};
```

The connection of this class to the CI `CFDSimu.ci` is then straightforward for classes whose methods match the interface. Some important features of the CTL are here used:

CTL::GROUP is the base for any CTL application. It keeps track of open communication channels, stores info for all group members and does the initial handshake. The CTL gives a size for the group that equals the number of MPI processes that one wants to instantiate a parallel run.

VECTOR<CFDSIMUPI> is a vector of `worker` where each component is an instance of `CFDSimuPI`. The number of `worker` corresponds to the number of MPI processes required for a parallel run.

CTL::RESULT is used to cope with received results of a call. Results are handled by the CTL. Here, the `ctl::result<int>` allows to receive the results of a solve in a non-blocking way. On the contrary, the master (`worker[0]`) is called in a classic blocking way as their results are carried out by classic instantiation of an integer.

The separation between the implementation of the method and its parallelization is again highlighted. Furthermore, any protocol supported by both CTL and OpenFOAM – MPI but also PVM – can here be used in this parallel version of the component.

3.3.3.2 *Parallel CFD Component Performances*

The performance of the parallel version of the ofoam-2 component is validated on the same problem of flow in a tube. For discretization the meshes are split via METIS [Karypis and Kumar, 1998] in subdomains with the same weight (Fig. 3.6).

For each computation, the bottleneck is the time where the evolution of the flow between two time steps is computed. The computational time measured in this section and in the two following ones is the average time T^{CPU} required to solve one iteration of the given problem and to write the associated results at the end of the time step. The initialization time is therefore not taken into account.

We consider two quantities to measure the quality of the parallelization; the speed-up χ measures the absolute clock time gain defined as:

$$\chi = \frac{T_1^{\text{CPU}}}{T_N^{\text{CPU}}} \tag{3.1}$$

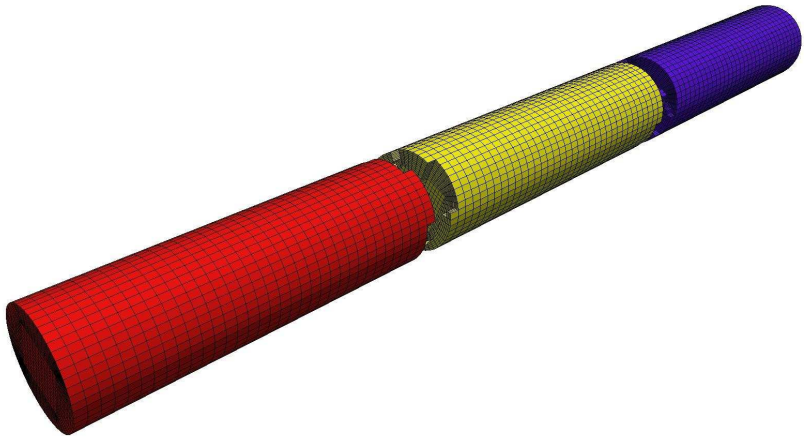
where T_1^{CPU} is the computational time for the problem solved on one processor and T_N^{CPU} for N processors. A linear speed-up cannot be obtained: it means that taking N processors, one cannot expect to divide the computational time by N. Indeed, the communication between processes, that is highly linked to size of interfaces, has to be taken into account. For this reason, the notion of efficiency can be introduced:

$$\xi = \frac{T_1^{\text{CPU}}}{N \times T_N^{\text{CPU}}} \tag{3.2}$$

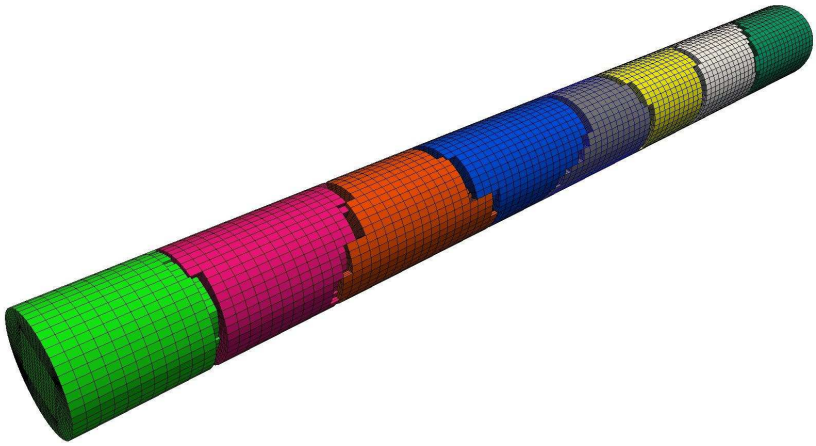
The parallel computations can be performed on the same multi-processor machine or on a cluster architecture where computers communicate throughout a network.

3.3.3.3 *Parallelization on the same multi-processor machine*

For this case we consider a rather coarse mesh with 72×10^3 cells ($\simeq 216 \times 10^3$ d-o-f). Computations are run with the same parameters as the one used for the comparison between ofoam and ofoam-2 (see Sec. 3.3.2.2). The results in terms of CPU time are given in Tab. 3.2.



(a) 3 sub-domains decomposition



(b) 8 sub-domains decomposition

Figure 3.6: Cylinder meshed with 9×10^3 cells split via METIS in sub-domains for parallel runs

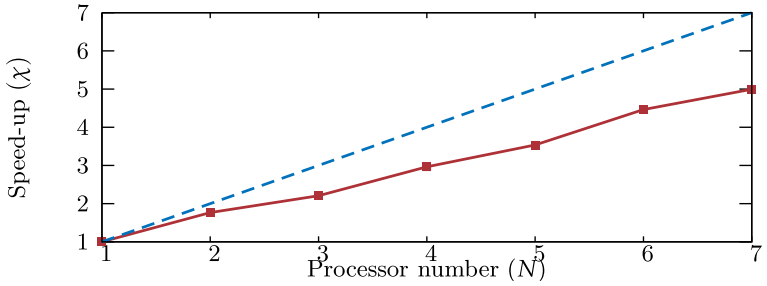


Figure 3.7: Parallel speed-up on a multi-processor machine

For this example, the results shown in Fig. 3.8 that efficiency of parallel

number of processor N	method `solve` T_N^{CPU} in (s)	method `init` T_N^{CPU} in (s)
1	5.18×10^0	9.57
2	2.92×10^0	10.75
3	2.34×10^0	10.17
4	1.74×10^0	9.52
5	1.46×10^0	9.24
6	1.16×10^0	10.15
7	1.03×10^0	9.86

Table 3.2: CPU Time for two methods handle by ofoam-2 component parallelized on the same machine with 2 Intel Quad cores at 3.0Ghz.

computing maintained around 0.7, even for 7 processors. Thus, in parallelization on 7 processors one can observe a Speed-up (Fig. 3.7) of around 5 times.

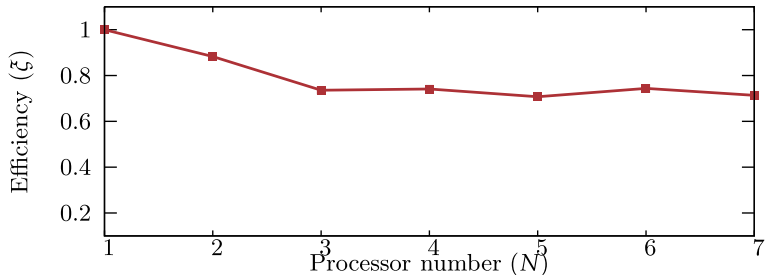


Figure 3.8: Parallel efficiency on a multi-processor machine

However, the question of the representativeness of this computation can be raised, as the serial case takes only 5.16s on one 3.0Ghz processor and, compared to that time, the communication cost between processes may not be negligible.

number of cells	reference T_1^{CPU} in (s)	compact transfer T_6^{CPU} in (s)	standard transfer T_6^{CPU} in (s)
9×10^3	3.91×10^{-1}	1.38×10^{-1}	4.52×10^{-1}
30375	1.54×10^0	5.27×10^{-1}	7.39×10^{-1}
72×10^3	5.18×10^0	1.16×10^0	1.27×10^0
243×10^3	2.84×10^1	8.14×10^0	6.97×10^0
576×10^3	1.07×10^2	4.13×10^1	3.17×10^1
4608×10^3	1.66×10^3	8.42×10^2	7.13×10^2

Table 3.3: CPU Time for the `solve` for different meshes 3.0Ghz with standard and compact transfert

For that reason, we compute the flow in a tube problem for different mesh refinements – from 27×10^3 to $\simeq 18 \times 10^6$ d-o-f – on one processor, and then repeat parallel computations on 6 processors. Here we present the computational time spent for the solve method with direct and compressed transfers between

the processes. In compressed transfer, each `double` is converted into a `float`, leading to a small loss of accuracy.

Those results are illustrated in Fig. 3.9. We represent there the efficiency of the parallel computation on 6 processors as a function of the mesh refinement. One observes a decrease in the performance for coarser grids, as for these cases the communication between processes is not negligible. This is even more emphsized when one chooses not to compress the `double` into `float`.

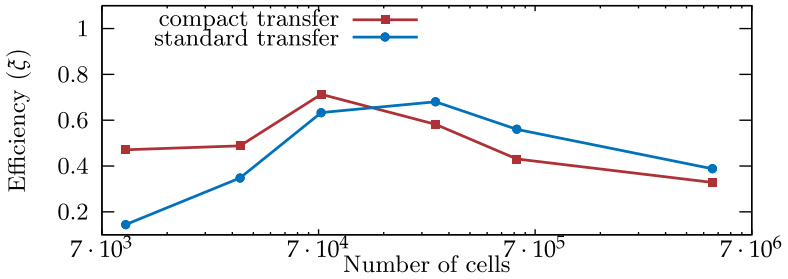


Figure 3.9: Parallel computation performance for different mesh sizes

For finer meshes, we also observe a decrease in the performance. Namely, imposing the incompressibility conditions in a parallel run requires more iterations than for a serial one. Indeed, it is known that state solver convergence is affected by running in parallel as the preconditioning and smoothing operations in the cells adjacent to the boundaries are less effective, and lead to a small increase in the number of iterations[2]. In those cases, decreasing the accuracy when the values between processes are transmitted is not a good choice: the gain obtained on the MPI communication process (especially on the same machine) does not justify the loss of precision and so forces an increase of the number of iterations.

Finally, for the finest meshes, the performance of the parallel computation drops significantly. This phenomenon seems to be due to the architecture of one cluster node where not only the processor velocity but also the bus communication speed for exchanging data between processors limit the overall performance, especially when huge transfer of data is required. This difficulty can be overcome with parallel computations on a cluster with several machines (see next section).

Note that for the cases in which the quantity transmitted between data is huge, special care has to be taken. Indeed, the CTL allocates the buffer size for MPI process with a default value that is quite small in order to minimize the memory requirement. This value naturally has to be increased when it is required.

3.3.3.4 *Parallelization on a cluster*

For the parallelization on several nodes (machines) of a cluster, a finer grid with 4.608×10^6 cells ($\simeq 18 \times 10^6$ d-o-f) is considered. Each node is linked with

[2]For more details, the reader is invited to see: http://www.cfd-online.com/Forums/openfoam-bugs/62430-difference-between-parallel

the other via a fast network, and possesses 8 Intel processors with a $3.0GHz$ frequency. Computations are run on eleven nodes of the cluster. The load of each node is maintained at the same level as much as possible. For instance, for the parallel computing on 32 processors, the first ten nodes were loaded on 3 of their processors whereas those on the last one two processors were used. The results obtained for the `solve` method are represented in Tab. 3.4. The time spent for other methods like initialization is not presented, as Tab 3.2 has shown that the parallelization is of little influence for these methods.

number of processor N	method `solve` T_N^{CPU} in (s)
1	1.55×10^3
2	7.10×10^2
4	3.85×10^2
8	2.33×10^2
16	1.31×10^2
32	9.64×10^1
64	8.26×10^1

Table 3.4: CPU Time for the `solve` method parallelized on 11 cluster nodes.

Fig. 3.10 represents the real speed-up observed to solve one time-step and the theoretical linear speed-up. An almost linear speed-up is observed until more than 4 processors of each node are loaded. The same value was observed in the previous section, when the parallelization was done on the processors of the same machine. This is probably due to some limitation of the architecture of the cluster nodes.

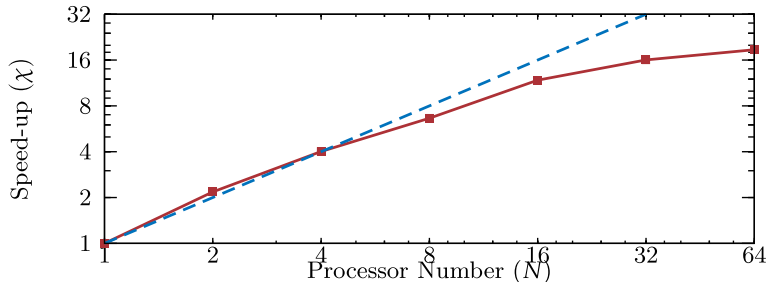


Figure 3.10: Parallel computation speed-up for runs on different cluster nodes

The efficiency of the parallel computing, as seen in Fig. 3.11 is therefore around optimum value of 1 before it starts decreasing. Note that the efficiency for parallelization on a small number of cluster nodes is above the theoretical optimum. This is due to the fact that less data need to be handled by the memory on each nodes.

It is also of interest to compute the CPU time for the same range of mesh size as the one choosen in Sec. 3.3.3.3. Tab. 3.5 gives the results obtained using the `CTL_Profile` tool that shows the characteristic times spent on each process of a CTL component.

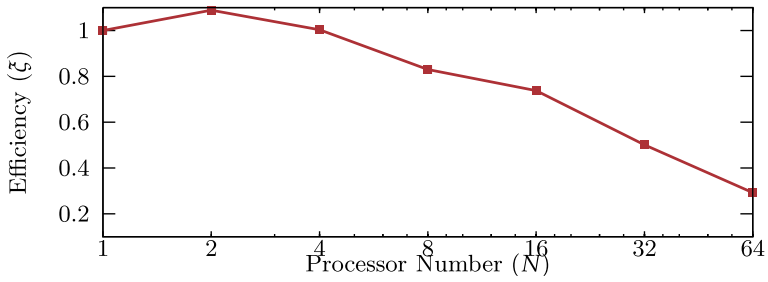


Figure 3.11: Parallel computation efficiency for runs on different cluster nodes

number of cells	reference T_1^{CPU} in (s)	standard transfer T_6^{CPU} in (s)	compact transfer T_6^{CPU} in (s)
9×10^3	3.83×10^{-1}	2.74×10^{-1}	3.33×10^{-1}
30×10^3	1.55×10^{0}	5.44×10^{-1}	6.38×10^{-1}
72×10^3	4.38×10^{0}	1.16×10^{0}	1.29×10^{0}
243×10^3	2.45×10^{1}	4.02×10^{0}	4.58×10^{0}
576×10^3	8.11×10^{1}	1.28×10^{1}	1.57×10^{1}
4608×10^3	1.55×10^{3}	2.59×10^{2}	2.90×10^{2}

Table 3.5: CPU Time for the `solve` for different meshes 3.0Ghz with standard and compact transfert

Also is represented the efficiency of the computation obtained for different mesh sizes (see Fig. 3.12). Here, each grid is split into 6 sub-domains, then solved on 6 different machines. Contrary to the phenomenon observed for parallel runs on the same machine (Fig. 3.9), the expected behavior is observed when increasing the number of d-o-f and parallelizing on different machines.

The bigger is the mesh, the more efficient is the parallel computation, since the communication between MPI processes then remains small compared to the time spent for solving the problem.

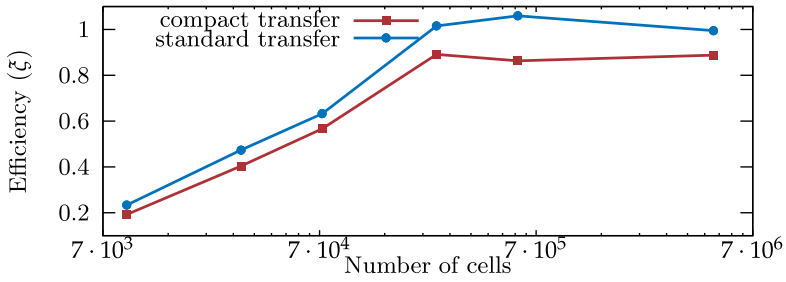


Figure 3.12: Parallel computation performance on six different cluster nodes for increasing mesh sizes from 216×10^3 to 18×10^6 d-o-f

For all computations of the studied flow case, handling compressed data from `double` to `float` for the exchanged values is not advantageous. For the fast communication between cluster nodes, the speed-up in MPI communication

is countered by the loss in accuracy that leads to more iterations to smooth the values and precondition the solvers, especially near the sub-domain boundaries.

3.4 The master code **co**ps

3.4.1 *Software coupling applied to fluid-structure interaction*

Coupling a fluid code and a structure code in order to solve fluid-structure interaction problem is often done in agreement with the coupling algorithm to be implemented. There are two main way to couple a fluid and a structure code in a partitioned way:

The master coupling approach. where a master code sends request and receive data from the coupled software. Traditionally this approach is quite intrusive, and imposes new developments inside the software. However, this problem is solved by the use of component technology, as described above, which makes modifications of the piece of software to accept the coupling and the development of the master code totally independent.

Another point to underline is the possible bottleneck effect at the master code. Namely, if sequential coupling with blocking calls for each component is not a major problem when one develops a software used on a single processor machine, special care has to be taken for parallel computing.

The autonomous coupling approach. where there is no master code, but the coupled codes directly communicate through an interpretor. It is therefore required to define a certain number of conventions that will make the dialogue between codes intelligible. Then the coupling algorithm is implemented in a dissociated way. At the end, a system equivalent to the piles and stacks of the processors will carry out the data received by each subproblem.

In [Garaud, 2008], the autonomous coupling approach is chosen in a fluid-structure interaction context. This approach is made possible by the use of a highly abstract `C++` code as the solid mechanics elementary brick (Zebulon [Foerch, 1996]). This idea is not easily applicable to functional codes programmed in `Fortran`. The main advantage of the autonomous coupling approach concerns a less intrusive (if the software can be coupled to a coupling engine library) and more easily communication processes for parallel computing (but this advantage is not clearly seen in [Garaud, 2008], since parts of the coupling engine were not supporting parallelism).

The use of component technology with the middleware CTL eliminates the difficulties in the Master coupling approach. We will here insist on two points: the development of a component based upon an existing code requires a very good understanding of its architecture, and a deep knowledge of the physics behind. Hence, once the component is developed, it can *actually* be used as a black box.

In the following, the master coupling approach is chosen, and the architecture of the master code **co**ps (for COupling COmponents by a Partitioned Strategy).

3.4.2 *Component architecture of **co**ps*

The COupling COmponents by a Partitioned Strategy (**co**ps) proposes a generic implementation of the explicit and implicit DFMT algorithms detailed in Chapter 2 for fluid-structure interaction. Note in passing that the same algorithm applies to any coupled problem with two sub-domains and an interface. The key idea is to re-use existing codes, such as `FEAP` or `OpenFOAM`, as elementary bricks to build upon the CTL components (see Sec. 3.3 for the fluid part and Sec. 3.2 for the structure part).

For the sake of generality, **co**ps itself is a component and has been successfully re-used, for instance in [Austruy et al., 2008]: a sequential multi-scale approach [Feyel and Chaboche, 2001] to solving fluid-structure interaction problem. Dependency graph of **co**ps component will not be detailed here, as the idea is really simple, but rather presented by a general overview of its architecture given in UML syntax in Fig. 3.13.

The interface, defined in a `.ci` file, is realized by a `C++` class named **co**ps. The main routine calls the coupling component, providing the path where the control file associated to **co**ps lies. The coupling component **co**ps instantiates two sub-solvers, the one of coFeap for the structure and another of ofoam-2 for the fluid problem and with a `partitionedSolver` binding the two subproblems. `PartitionedSolver` is a template class that couples any component matching a `SimuCI` or a `CFDSimuCI` where method is necessary to build the Steklov-Poincaré operators, and optionally `goback` in time if sub-iterations of one subproblem in implicit computations is asked.

The `PartitionedSolver` instantiates a `Picard` method that is in charge of one Picard iteration of the coupled subproblem (see Eq. (2.32)). One Picard iteration allows to determine the new residual. Thus the computation will continue convergence of a BGS solver. The Picard class itself relies on a template SteklovPoincare class. Improving the solver can be done adding a Schur class that proposes to compute the Jacobian of each subproblems.

3.4.3 *Field interpolation between solvers*

The use of different different solvers, for the fluid and the structure part, do not provide in general a matching mesh at the interface. Furthermore, even for matching meshes, as the geometries of the domains are not the same on both sides of the interface, an optimal numbering of the nodes can lead to different orders for the interface nodes. Last but not least, different discretization techniques (Finite Element versus Finite Volume) or different order p of the polynomials can be used for constructing solution to fluid-structure interaction problem. In the domain of FE applied to mechanical engineering, extensive literature can be found on how to build a consistent interpolation for both subproblems at the interface [Felippa and Park, 2004]. For the fluid-structure interaction problems, an interesting review can be found in [de Boer et al., 2007]. For the sake of simplicity, the use of Dirac functions has sometimes been proposed [Ibrahimbegović and Markovič, 2003, Hautefeuille, 2009].

In our framework, it was decided not to favor any mesh-based representation of the interface, since the fluid problem can also be solved by a meshfree-based

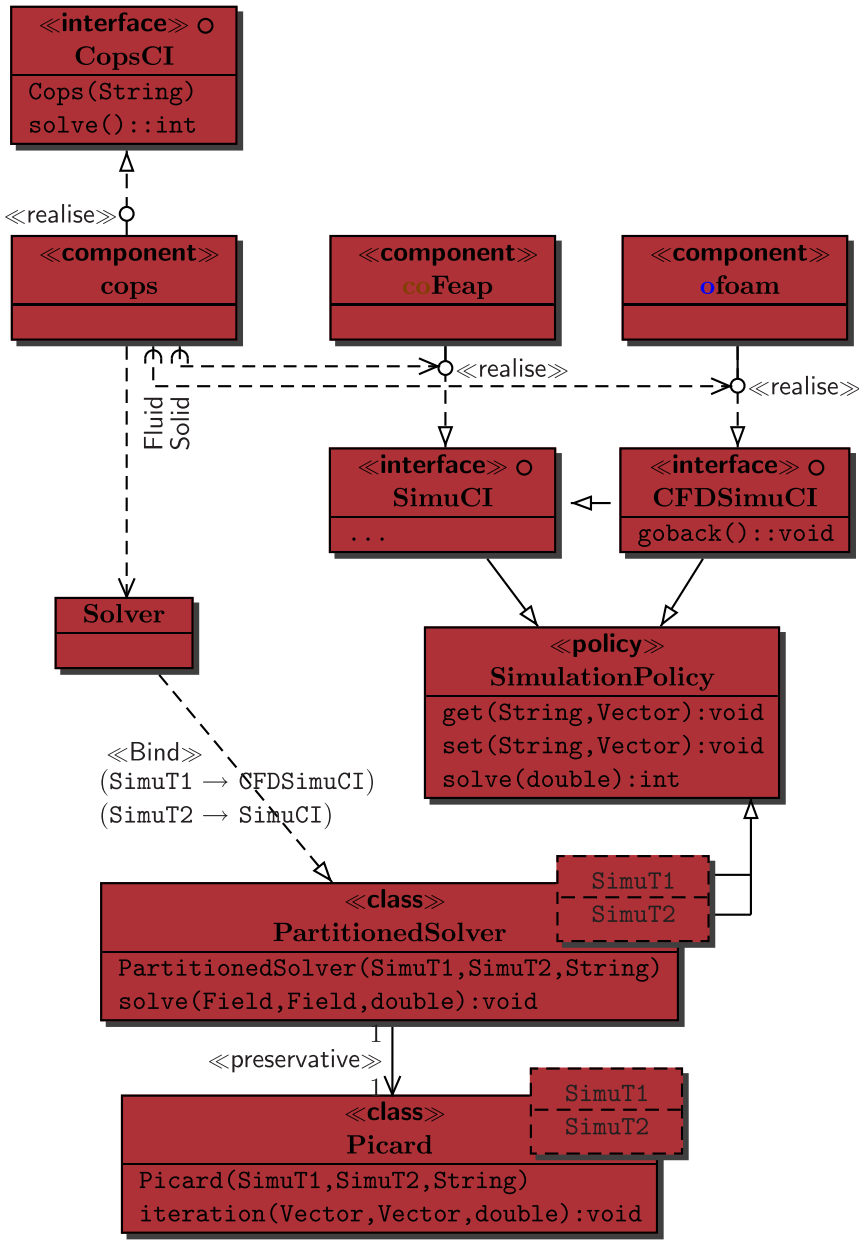


Figure 3.13: A simplified UML class diagram for **co**ps

method [Fries, 2005]. Namely, an interpolation strategy relying on radial basis function is here chosen [Beckert and Wendland, 2001]. This fits well with the current work where, both sides of the problem are based on different approximation methods. First, for the structure we employ the FE solver which requires values that are imposed at the nodes and gives in return nodal values

as answers. Second, in the FV discretization for the fluid part, the fields at the interface are defined at the center of the cells, and not at the nodes as it is done for FE. However, as the client has no reason to know the connectivity table of the coupled component nor the faces centers coordinates, it was decided to interpolate the fields internally in ofoam-2 and to set or get them at the points of the interface.

The Steklov-Poincaré operators for the fluid and the structure request imposed boundary values at the nodes placed interface, and provide the values, for potentially different nodal points $\boldsymbol{x}_f$ and $\boldsymbol{x}_s$. The interpolation step from fluid to structure consist in solving a linear system as explained in the following.

Let us consider that the displacement at the N_s solid nodes $(\boldsymbol{x}_{s,i})_{i=1...N_s}$ is given as $(\boldsymbol{u}_{s,i})_{i=1...N_s}$. We want to build the interpolation of this displacement field at the N_f fluid nodes in order to impose the fluid mesh motion as needed in ALE fluid computations. An interpolant for each scalar field of the projected displacements in one of the Cartesian directions has then to be built. This scalar field is denoted as $u_i = u(\boldsymbol{x}_{s,i})$ at each points, and the interpolation has to be performed d times for each vector field of dimension d.

Then the interpolant at a point $\boldsymbol{x}$ takes the form:

$$u(\boldsymbol{x}) = \sum_{i=1}^{N_s} c_i \Phi(\boldsymbol{x} - \boldsymbol{x}_{s,i}) \tag{3.3}$$

where Φ is a fixed basis function which is radial with respect to the Euclidian distance, *i.e.*:

$$\Phi(x) = \Phi(\|\boldsymbol{x}\|_2) = \Phi\left(\sqrt{\sum_{i=1}^{d} x_i^2}\right) \tag{3.4}$$

The coefficients c_i are determined by the interpolation condition:

$$u_j = \sum_{i=1}^{N_s} c_i \Phi(\boldsymbol{x}_{s,j} - \boldsymbol{x}_{s,i}) \tag{3.5}$$

Thus, for interpolation of any field to interpolate, the coefficients c_i have first to be computed by inverting the following relation:

$$M_{ij} c_j = u_j \tag{3.6}$$

where the interpolation matrix entries are computed as:

$$M_{ij} = \Phi(\boldsymbol{x}_{s,j} - \boldsymbol{x}_{s,i})$$

The last step is to compute the interpolated field by using the expression given in Eq. (3.3).

The choice of the radial basis function is governed by the requirement that the influence of a center has to be smaller when the distance to an evaluation node increases. Elsewhere, the global character of the radial basis function tends to smooth out all local effects. Another desirable property is to use the functions with compact support which results with a sparse interpolation matrix.

In [Beckert and Wendland, 2001], a comparison between different smooth compact radial basis functions for field transfer at the interface is given. In this work, global radial basis function are used in the exponential form:

$$\Phi(x) = \exp -\frac{\|x\|}{h} \tag{3.7}$$

where h is a characteristic distance between points.

The `Interpolator` component was implemented by D. Jürgens [Jürgens, 2009] from the TU-Braunscwheig. The advantage of the component use is the possibility to plug-in any interpolation software to the fluid-structure interaction framework that matches the Component Interface. The inverse problem of the full matrix M_{ij} is here relying on the `lapack` or the Intel Math Kernel Library; the cost of such an operation is around N^3, but as the interface d-o-f are by far less than number of d-o-f for each subproblem, this cost cannot be the bottleneck of the computations. One of the advantages of the proposed approach is that the interpolation is exact for matching nodes. It can be used without any lack of precision in a generic way for matching meshes where the ordering is not the same for both subproblems.

Closure

In this Chapter, a general fluid-structure interaction framework based on existing software was presented. This framework was built using the middleware CTL which offers good performances, and can therefore be used for scientific computing of large systems. From the point of view of software engineering, the following properties of the components where used:

Multi-usability: several components can be instantiated in order to perform parallel computation. The building of a parallel component for the fluid part based on the existing paralleling feature of a CFD code was explained.

Non-context specific implantation: two components fitting the same interface can be used in the framework proposed. For instance, a structure component based on Parafep as well as a fluid component based on files reading and system calls for OpenFOAM can be plugged in **co**ps.

Composability: **co**ps, that handles the coupling of components, can be used as a component in a more general framework computation [Austruy, 2008].

Encapsulation: the inner structure of the component is not accessible. That allows to separate algorithms from their implementation and to implement components independently from the communication method.

Another important feature is the possibility to couple software products that were initially not programmed to be coupled, even if they are based on different discretization techniques (respectively FV for the fluid and FE for the structure) and were programmed in different languages `C++` and `Fortran`. The possibility to re-use components developed for other contexts is also emphasized: coFeap was used to build a multi-scale FEM-based code MuscaD [Hautefeuille, 2009], and the `Interpolator` was not initially built for this particular context.

4 Fluid-Structure interaction numerical examples

In this chapter the fluid-structure interaction framework previously presented is applied to solve some numerical cases. The first part of this part deals with the validation of the presented implementation on a an academic case. The artificial "Added-Mass effect" and its cure using implicit solvers is experimentally observed. Then, long terms runs are presented for both two and tri-dimensions. The parallelization of the fluid part is used. Finally, the interaction between a structure and a free surface flow representing the breaking of a wave is explored.

Contents

4.1 Driven cavity with flexible bottom 95
4.1.1 The lid-driven cavity fluid problem 95
4.1.2 Modification for the FSI validation case 96
4.1.3 Subproblems discretization and numerical parameters 99
4.1.4 Tight coupling with the modified lid-driven cavity case 99
4.1.5 Implicit strong coupling and reference solution 99
4.1.6 On the impossibility to apply explicit coupling 101
4.2 Performances and fluid domain decomposition . . 104
4.3 Flexible appendix in a flow 106
4.3.1 Problem description 106
4.3.2 Implicit coupling and reference solution 109
4.3.3 Explicit coupling 110
4.4 Three-dimensional flag in the wind 112
4.4.1 Problem description 112
4.4.2 Fluid discretization 113
4.4.3 Solid discretization 114
4.4.4 Coupling 114
4.4.5 Results 115
4.5 Two-dimensional wave hitting a structure 116
4.5.1 Introduction and a first approach 116
4.5.2 Problem description 117
4.5.3 Fluid discretization 117
4.5.4 Structure discretization 118
4.5.5 Coupling 118
4.5.6 Results 119
4.6 Three-dimensional wave impacting a structure . . 121
4.6.1 Problem description 121
4.6.2 Fluid discretization 121
4.6.3 Solid discretization 122
4.6.4 Coupling 122
4.6.5 Results 123
Closure 125

Building a software environment for fluid-structure interaction problems over existing computational codes requires, as in any scientific computing development, testing and validating the environment. One of the major problems encountered is to design sufficiently simple FSI problems. For instance, we started by trying to validate the environment with the example of a rigid cylinder oscillating in a fluid-flow [Dettmer and Perić, 2007], and thought the problem was simple enough to identify the source of mistakes during the development. After many trials, it appeared that the problem was not simple enough to distinguish the different sources of error.

After some trials, it becomes clear for that the lid-driven cavity problem with a flexible bottom [Wall, 1999] is a really good way to validate an fluid-structure interaction strategy ... when the boundary conditions are properly defined (see Sec. 4.1). This problem shows the convergence properties of the implemented DFMT-FSI algorithm with dynamic relaxation, and illustrates the impossibility to use an explicit coupling method due to the added-mass effect. Furthermore, it allows to validate the coupling with a parallel version of the fluid component.

In the following, the application of the coupling environment for two distinct flow regimes is presented. The models here chosen have an academic geometry. They are simplified representation of long term computation suitable to represent structures interacting with wind, and structures impacted by sloshing waves.

Long run flutter examples: (Sec 4.3 and 4.4) Two- and three-dimensional problems of thin structures interacting with vortices shedded behind a rigid bluff body . The two-dimensional problem presented herein, first introduced in [Wall and Ramm, 1998] is widely used as a validating example (see [Mok, 2001, Matthies and Steindorf, 2003, Hübner et al., 2004] and [Dettmer and Perić, 2007]). An FSI example in a three-dimensional space was recently introduced [von Scheven, 2009]. It allows to show the development capacity in terms of expensive computation with an important number of d-o-f and many time steps.

Sloshing wave hitting a structure: (Sec 4.5 and 4.6) It is an original application when coupling an FV CFD solver with a FEM structural one. A few examples can be found in the literature [Walhorn, 2002], and they are based on monolithic approaches or use different fluid-strategies. A fully three-dimensional computation for this range of application is also presented.

4.1 Driven cavity with flexible bottom

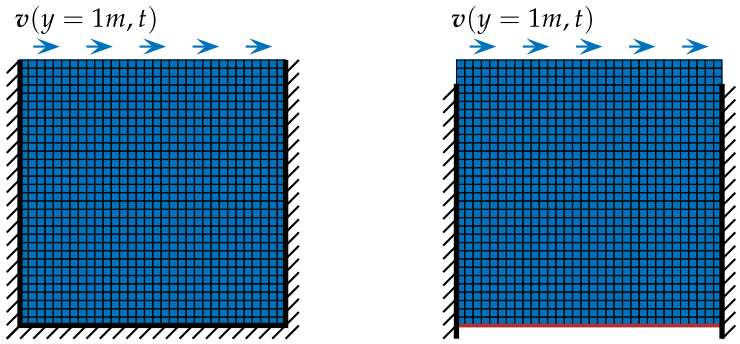
Having recognized the value of this problem for establishing the standard benchmark in FSI, a sufficiently detailed presentation of the problem and results obtained is provided in order to promote the comparison.

4.1.1 *The lid-driven cavity fluid problem*

The driven cavity consider 2D fluid flow problem in a square domain. The imposed boundary conditions at three out of four sides are the zero value of velocity.

Only at the top of the cavity has a nonzero velocity imposed. The geometry and boundary conditions are also depicted in further details in Fig. 4.1(a).

The flow is governed by Navier-Stokes equations, with a dominant convective term, which is usually the most difficult to solve with accuracy. Furthermore, when the imposed velocity is sufficiently small, the flow is laminar and incompressible. In fact, the driven cavity is traditionally solved as a validation example for CFD codes, especially for its convective dominance features. Therefore extensive literature is dedicated to its study, ever since the early works in computational fluid dynamics [Ghia et al., 1982] to more recent reviews [Bruneau and Saad, 2006] using different fluid solvers [Hortmann et al., 1990, Ferziger and Perić, 2002]. It is also proposed as a tutorial case for the fluid solver `OpenFOAM` [OpenCFD LTD, 2009] employed herein. In the present test, the cavity is discretized by finite volume 32×32 FV cells.



(a) Lid-driven fluid flow in cavity (b) Modified lid-driven cavity for FSI

Figure 4.1: The lid-driven cavity example

4.1.2 *Modification for the FSI validation case*

For the case of fluid-structure interaction benchmark, introduced for the first time in [Wall and Ramm, 1998], the flexible membrane is placed at the bottom boundary (see Fig. 4.1(b)). The material properties are chosen as follows: the flow is solved with Navier-Stokes equations and we consider a Saint-Venant Kirchoff material able to undergo finite deformation for the solid part. The density for the fluid is $\rho_f = 1kg \cdot m^{-3}$ and the kinematic velocity $\nu_f = 0.01m \cdot s^{-2}$. The structure is really soft, with a Young modulus $E_s = 250Pa$, no Poisson ratio $\nu_s = 0$ and a density of $\rho_s = 500kg \cdot m^{-3}$.

For the lid-driven cavity flow in Fig 4.1(a), all boundary nodes constraint by Dirichlet condition on velocity, and pressure field remains undetermined up to a constant. If this does not lead to any problem when only fluid flow is considered with no interaction with the structure, a special care has to be taken for the fluid-structure interaction case where the imposed velocity condition should satisfy incompressibility condition and where the exact value of pressure is needed to

define structure boundary conditions. It is studied in [Küttler et al., 2006].

Fluid-structure interaction problems with pure Dirichlet boundary conditions are subjected to the so-called incompressibility dilemma. Therefore, none of the partitioned DFMT strategies presented previously – neither explicit nor implicit – will succeed to solve such a problem. Three remedies are proposed in [Küttler et al., 2006]:

VOLUME CONSTRAINT APPLIED TO THE STRUCTURE EQUATIONS OF MOTION: an additional volume constraint of the fluid domain equations is added to traditional structure equations.

NEUMANN-DIRICHLET PARTITIONED: the traditional Dirichlet-Neumann DFMT partitioned strategy, in which primal fields from the structure and dual ones from the fluid part are exchanged, is inverted; and, the structure is moved under imposed velocity and the fluid boundary is solved at imposed pressure.

ARTIFICIAL COMPRESSIBILITY: The incompressibility constraint is temporarily removed from fluid computation. Hence an algorithmic compressibility is introduced.

One way to overcome this difficulty in the fluid-structure interaction lid-driven cavity problem, and not to consider such subtleties, is to release two nodes on each size of the cavity (see Fig. 4.1(b) and [Wall, 1999]). A subtle point to be mentioned concerns the chosen discretization techniques: the stabilized FE approximation used in [Wall, 1999, Küttler et al., 2006] allows to remove constraints at the nodes, whereas the FV method used herein requires that the corresponding velocity values at the boundary face of cells be left free. Another modification of lid-driven cavity for fluid-structure interaction concerns the time-dependent velocity boundary condition defined as:

$$\boldsymbol{v} \cdot \boldsymbol{e}_x = 1 - \cos\left(2\pi \frac{t}{T_{\text{char}}}\right)$$

where $T_{\text{char}} = 5s$. For such harmonic function, the solution of the fluid and of the fluid-structure interaction problem will exhibit an oscillating behavior, after a short transition period. At its maximum, the Reynolds number in the cavity reaches $\mathcal{R}e = 200$, and thus the flow can be considered as laminar.

This example has a great potential of becoming a perfect benchmark test for the following reasons:

MESH SIMPLICITY: as the Reynolds number is not more than $\mathcal{R}e = 200$, a 32x32 cells with a second order FV solver is sufficient to get an accurate solution [Ghia et al., 1982, Hortmann et al., 1990].

COMPUTATIONAL TIME: due to the small mesh size, computations are fairly inexpensive, especially for a fluid-structure interaction problem. Indeed, for each time step of $\Delta t = 0.1s$, the flow computation takes $1.08 \times 10^{-1}s$ and the solid one $2.95 \times 10^{-3}s$ when run on a single processor.

HARMONIC SOLUTION: the problem quickly reaches a harmonic solution, which provides a perfect platform to test the energy creation or dissipation in staggered or iterative fluid-structure interaction algorithms.

We thus arrive to a problem where only boundary conditions of the velocity field are imposed, and there are no condition on the pressure field p. In the vocabulary of CFD, these boundary conditions are called zero gradient (or Neumann zero). Thus, any constant can be added to the pressure field computed. This is not a difficulty when considering a only fluid sub-problem, since the gradient of the pressure field (see Eq. (1.13)) will filter out any constant pressure. It can lead to an non-unique solution of the pressure, and most of the iterative solver will fail for such a problem if no remedy is proposed. Furthermore, this no longer applies to fluid-structure interaction, where exact pressure is needed to impose the structural motion.

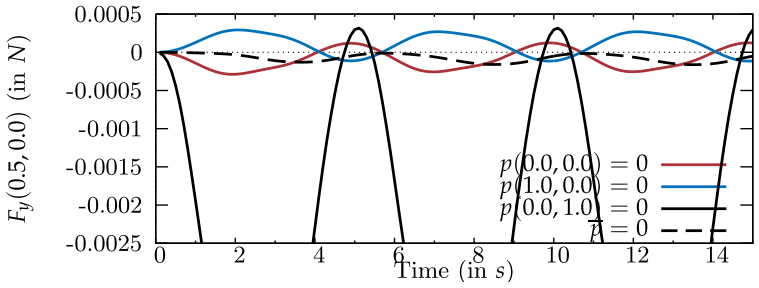


Figure 4.2: Force acting at the center of the flexible bottom for some fixing pressure strategies

As noted in [Ferziger and Perić, 2002] for the fluid flow case, there are two ways to specify a unique pressure field when only zero gradient boundary conditions are specified. For example, the pressure in a arbitrary cell is fixed (for instance to zero), or one can consider a pressure field with the mean value equal to an imposed constant.

For the present version of the lid-driven cavity dealing with fluid-structure interaction problem, the issue of pressure computation becomes more stringent since not only the gradient of the pressure field, but also its actual must be known at the interface. Unfortunately, the imposed condition on the pressure field are not clearly specified, neither in the first work [Wall and Ramm, 1998], nor in the following works of the same team [Mok, 2001, Küttler et al., 2006] or other teams [Gerbeau and Vidrascu, 2003b]. In [Bathe and Zhang, 2009], the pressure field is imposed at the flow input and output, the obtained result is totally different from the one given by the previous works predicting a high pressure at the bottom of the cavity leading to an average negative displacement of the structure in the $\boldsymbol{e}_y$ direction, and requiring the use of a stiffer material in order to remain in the acceptable displacement range.

In order to illustrate the importance of the chosen pressure condition in fluid-structure interaction analysis we take the case of a very stiff (potentially rigid) structure at the bottom and compute the total force (from pressure and viscosity) applied to its center (nodal force at $(0.5, 0.0)$). All these results are

presented in Fig. 4.2, showing quite different time evolutions of pressure force on the structure for different choices of pressure conditions. In the following, the pressure was fixed to zero at the out flow. Other boundaries are considered to be Neumann zero for the pressure.

4.1.3 *Subproblems discretization and numerical parameters*

The fluid domain is discretized with fourth order FV in space, and the time integration is handled by implicit Euler. The momentum equation is solved by PBiCG, with a DILU preconditioner whereas the pressure correction problem is worked out by iterative PCG, with DIC preconditioner. Two iterations of the PISO correction algorithm are performed at each time step, and the precision required for the iterative algorithms for pressure and velocity is 10^{-8} As the mesh is initially orthogonal, there is no reason to require initially non-orthogonal correctors. However, as the deformation of the bottom will eventually produce non-orthogonal meshes, it is specified that two non-orthogonal corrections should be performed at each time step.

Indeed, the fluid domain is then subject to structure imposed motion as the bottom of the domain (assuming no cavitation takes place and fluid follows the deformation of the structure), and the mesh deformation is handled by a smoothing process. The Laplacian equation is solved with the coefficient that depends on the distance to the bottom. On vertical walls, the points are allowed to move vertically ; elsewhere, the boundary nodes are fixed.

For the structure FE model a spatial discretization with $16\times$Q8 (quadratic) elements is chosen. The time integration is carried out by a implicit generalized α-HHT schemes, with a zero spectral radius as in [Förster et al., 2007], in order to maximize numerical damping. The non-linear algebraic equations are solved at each time step by the Newton iterative algorithm with a prescribed tolerance of 10^{-8}. A non-symmetrical direct solver is used at each iteration.

4.1.4 *Tight coupling with the modified lid-driven cavity case*

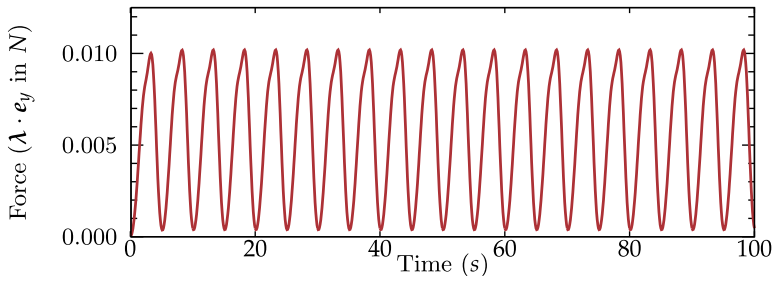
In Fig 4.3 the displacement is displayed when no interaction is considered. Namely, the forces are computed by the CFD-based component, and applied to the structure, but the structure motion is not imposed on the fluid domain, and thus there is no need to compute any mesh motion. The pressure value is fixed to zero at the outflow boundary.

4.1.5 *Implicit strong coupling and reference solution*

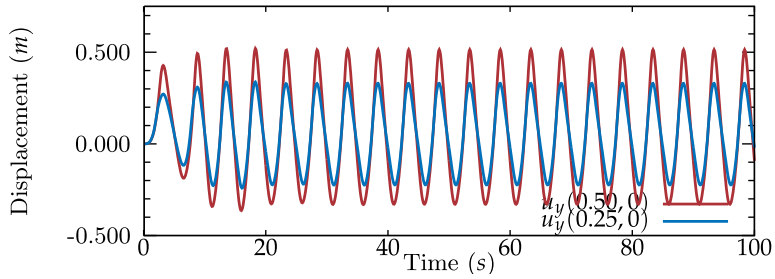
The computation is run with a coupling window size of $\Delta t = 0.1s$ and the coupling scheme used is DFMT-BGS either with fixed or Aitken's relaxation. The absolute tolerance considered is:

$$\|\boldsymbol{r}_N^{(k)}\|_2 \leq 1 \times 10^{-7}$$

The typical results for the pressure field, streamlines and domain deformation are plotted in Fig. 4.4.



(a) Force applied on the structure at the middle of the interface: $F_y(0.5, 0)$.



(b) Displacements at the fluid-structure interaction interface at the middle and the quarter lengths: $u(0.5, 0)$ and $u(0.25, 0)$.

Figure 4.3: The lid-driven: forces applied at the bottom of the domain and displacement obtained from their application to the the flexible domain thanks to a weak coupling strategy

The same kind force and displacement at the center of the flexible bottom are presented in Fig. 4.5 The results largely differ from the one obtained with a weak interaction model (see Fig. 4.3). In the strong coupled case, the inertia of the flexible bottom leads to a positive mean deformation. This deformation of the fluid domain induces a large decrease in the pressure and accordingly in the force amplitude.

Note that the displacement remains exactly the same for all predictors and relaxation techniques, once convergence to a residual norm less than 10^{-7} is obtained. The results in term of displacement are also compared to the one obtained in [Wall, 1999] and [Gerbeau and Vidrascu, 2003b] (see Fig. 4.5(c)).

Note that the inertia of the flexible bottom leads to a positive mean value of structure displacement. Moreover, this further induces a large decrease in the pressure and accordingly in the force amplitude. The characteristic solution is represented in Fig. 4.4, with the domain deformation, streamlines and pressure field.

With a fixed under-relaxation of $\omega = 0.25$, a constant decrease of the residual with a low order is observed (see Fig. 4.8). Aitken's relaxation allows to reduce the mean number of iterations required from 30 to 17 (see Fig. 4.6).

Aitken's relaxation does allow to improve the order of convergence. The convergence exhibits a less smooth behavior, with a larger decrease of the residual when the relaxation parameters increase (for instance the 4^{th} iteration in

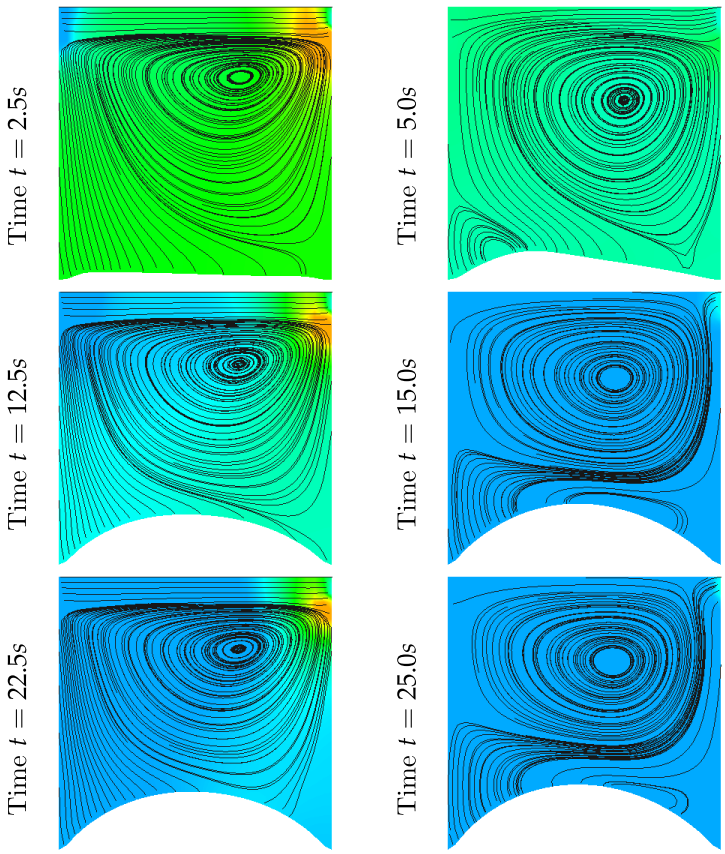


Figure 4.4: Lid driven cavity with a flexible bottom: snapshots with pressure field and streamlines for different time steps

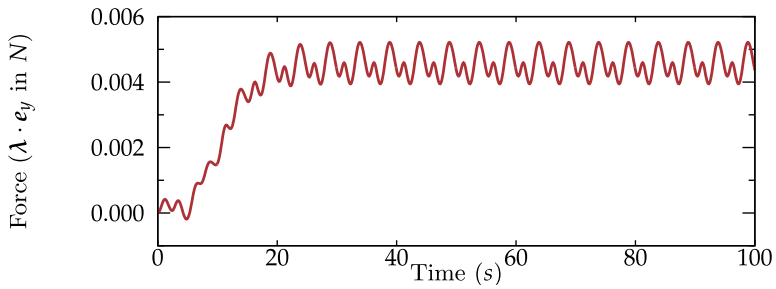
Fig. 4.10(a) and 4.8(a)). The relaxation parameter value is given in Fig. 4.10 for two chosen times $39.0s$ and $41.0s$ used in Fig. 4.8 to represent the convergence of the residual; The characteristic oscillations depicted in [Küttler and Wall, 2008] are observed.

The use of predictors does not change the order of convergence, but reduces the initial error in terms of residual (see Fig. 4.9). Thus, the characteristic number of iterations to reach required precision decreases from 17 iterations for a zero order to 7 for a second order predictor (Fig. 4.7).

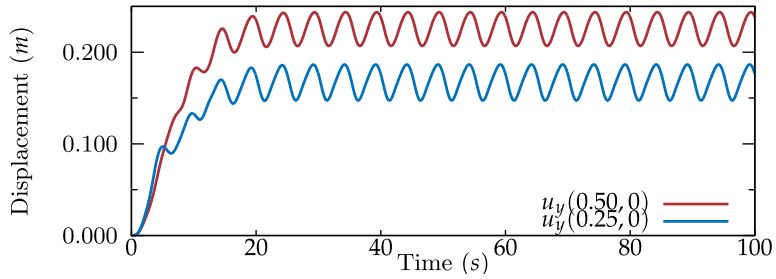
4.1.6 *On the impossibility to apply explicit coupling*

At the end of this section the results are presented for explicit DFMT coupling algorithms applied to the lid-driven cavity with flexible structure at the bottom. The added mass effects characteristic of explicit algorithms for the case of an incompressible flow is observed in Fig. 4.11, with each computation sooner or later diverging

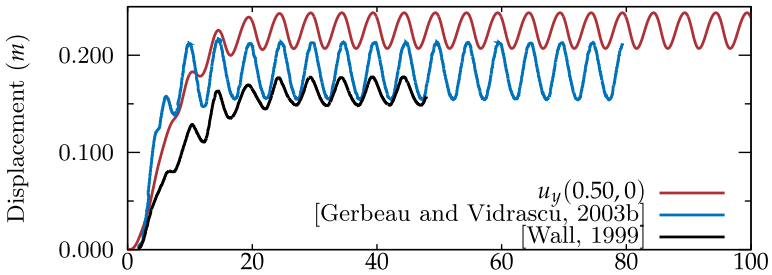
Our result divergence was not in agreement with the results convergence



(a) Force applied on the structure at the middle of the interface: $F_y(0.5, 0)$.



(b) Displacements at the fluid-structure interaction interface at the middle and the quarter lengths: $u(0.5, 0)$ and $u(0.25, 0)$.



(c) Displacement of the flexible bottom center compared with results from [Wall, 1999] and [Gerbeau and Vidrascu, 2003b]

Figure 4.5: Implicit computation of the lid-driven cavity with flexible bottom solved with BGS and relaxation.

presented in [Förster et al., 2007] for the least accurate predictors. The main reason, we believe, is due to the way the enforcement of the incompressibility condition for the fluid problem is performed (with FV and PISO algorithm used herein to indeed enforce this condition versus stabilized FE discretization used in [Förster et al., 2007]).

In [Förster et al., 2006], parameters are recognized as important for triggering earlier instability of explicit coupling:

Predictor order: (see Fig. 4.11(a)) the higher the order of the predictor, the sooner the instability occurs.

Window size for synchronization: (see Fig. 4.11(b)) for small coupling

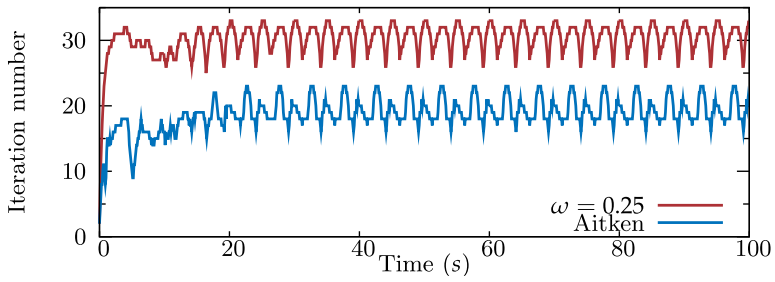


Figure 4.6: Number of iterations per time step for the BGS solver with fixed and Aitken's relaxation (order 0 predictor)

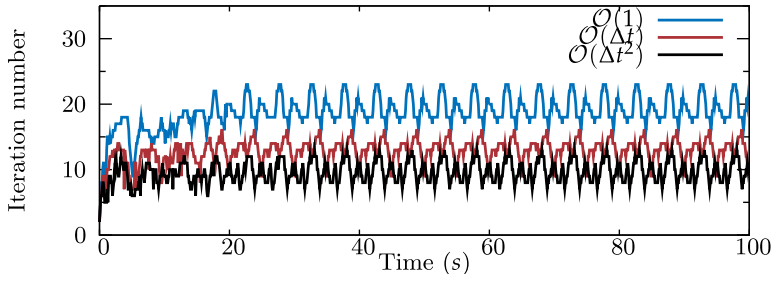


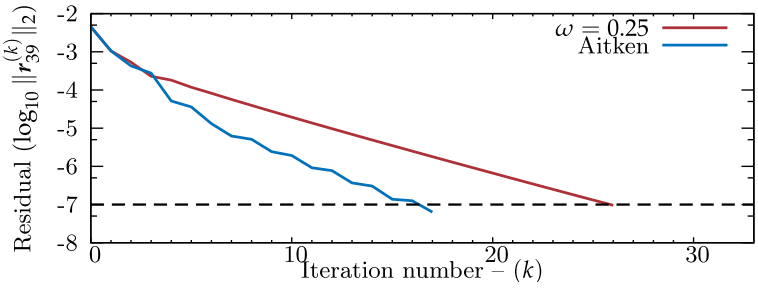
Figure 4.7: Number of iterations per time step for the BGS solver with predictor of order 0, 1 and 2 (Aitken's relaxation)

time steps, the divergence of the explicit coupling algorithm quickly occurs.

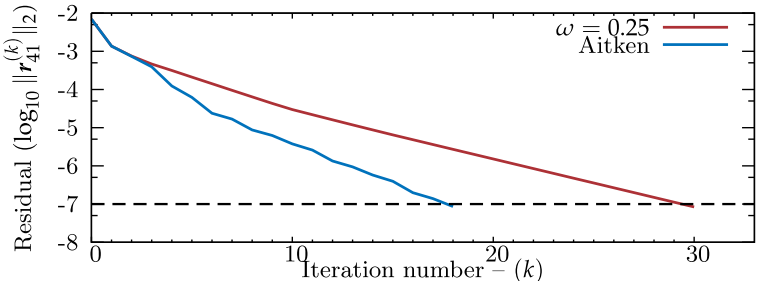
ORDER OF FLUID TIME INTEGRATION: (see Fig. 4.11(c)) More precise fluid integration (order of the time integrator for the fluid subproblem or time steps size when sub-cycling) degrades the stability behavior of the explicit coupling scheme.

NON COLLOCATED ALGORITHM: (Fig. 4.11(d)) Non-collocated algorithms (like DFMT-ISS) are more sensitive to the added-mass effect than collocated ones (DFMT-CSS). The use of correctors for the force does not improve the sensitivity collocated and non-collocated algorithms.

In accordance to our theoretical developments, it is clear (and has been confirmed by numerical examples) that the added mass effect also depends on: Mass ratio between fluid and solid with larger ρ_f/ρ_s, the instability occurs earlier; larger fluid velocities also trigger and more flexible structure instability more quickly. The phenomena observed are in agreement with the theoretical observations made in the App. A and in [Causin et al., 2005, Förster et al., 2007]. The general conclusion is that accurate time integration of the subproblem and coupling algorithms are more sensitive to the artificial "Added-Mass Effect" and diverge more easily.



(a) Time $t = 39.0s$



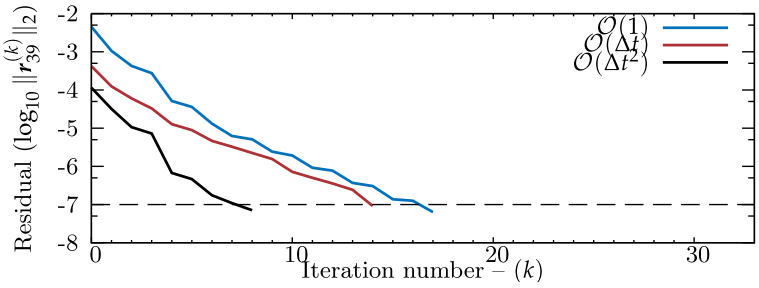
(b) Time $t = 41.0s$

Figure 4.8: Residual convergence for fixed and Aitken's relaxation with order 0 predictor

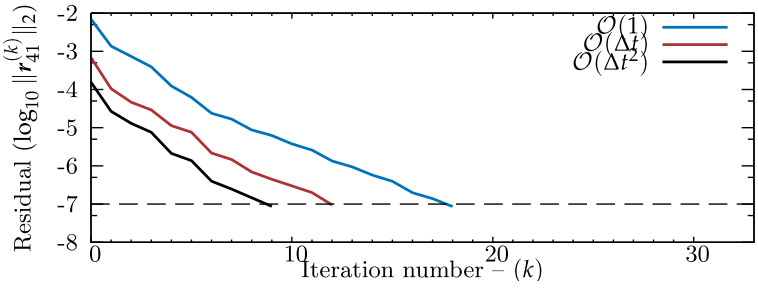
4.2 Performances and fluid domain decomposition

The lid-driven cavity displays the typical order of magnitude for the computational costs of the fluid and structure parts, as well as the interpolation between fields for a problem of the "flow around a structure" kind. Looking into further details, the `CTL_Profile` tool gives the distribution of the computational time for each component. Hence, in the first $25s$ of the computations, with a second order predictor and Aitken's relaxation, the results given in Tab. 4.1 are obtained. Most of the time is spent on the fluid part, which justifies the importance accorded to the inner parallelization of the fluid component. Also note that more than 250 calls are required to solve the first $25s$ with a time step of $0.1s$ as the implicit scheme requires some iterations. Finally, the Interpolator is called three times at each iteration, since the displacements and velocities (even if those are not required by the fluid component) from the structure as well as the forces at the interface are translated. Also note that the number of iterations required to reach the convergence criterion is slightly different for the standard and the parallel versions.

Fig. 4.12 represents the deformation of the fluid domain and the stream lines over the pressure field. The splitting of the fluid domain by METIS can lead (as it is the case here) to intersecting domain decomposition interface and fluid-structure interface. The new points (two points with the same coordinates in 2D, but it can be more in 3D) are naturally handled by the Interpolator. For the fluid to structure interpolation, the values from the coinciding points are added



(a) Time $t = 39.0s$



(b) Time $t = 41.0s$

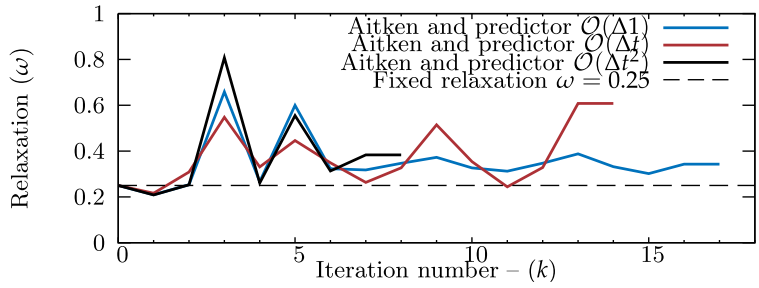
Figure 4.9: Residual convergence with Aitken's relaxation and predictor of order 0, 1 and 2

Component	Time (s) by call	Number of calls	Total time (s)
Fluid solved on one processor			
`Interpolator`	1.77×10^{-4}	2586	0.46
coFeap(solid)	2.95×10^{-3}	862	2.54
ofoam-2 (fluid)	1.08×10^{-1}	862	93.20
Fluid solved in parallel on two processors			
`Interpolator`	1.83×10^{-4}	2574	0.47
coFeap(solid)	2.96×10^{-3}	858	2.54
ofoam-2 (fluid)	8.27×10^{-2}	858	71.00

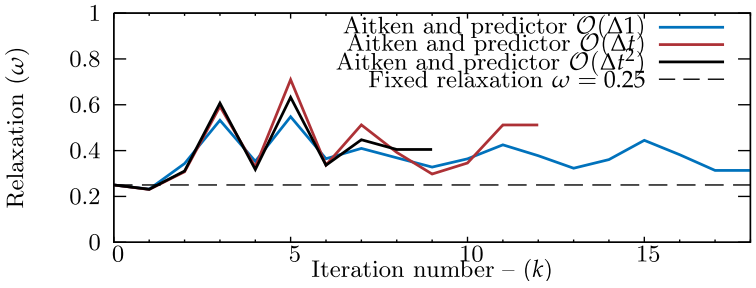
Table 4.1: Performance comparison for the first 25s of simulations with and without parallel run of the fluid flow.

to build the interpolated field. They are naturally interpolated in accordance with the number of coinciding points.

The efficiency of the paralleling for a problem with so few d-o-f on the fluid part is far from being satisfying. Indeed, as explained in Sec. 3.3.3, for such a small problem, the time spent on communication can compare in magnitude with the time spent on computation.



(a) Time $t = 39.0s$



(b) Time $t = 41.0s$

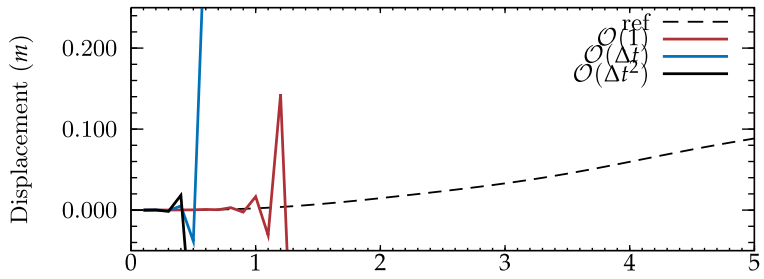
Figure 4.10: The lid-driven: evolution of relaxation parameter for two characteristic time steps

4.3 Flexible appendix in a flow

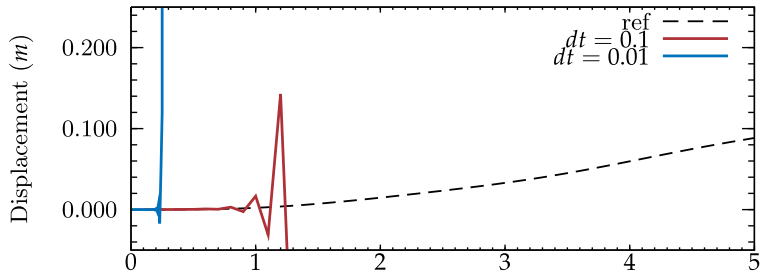
4.3.1 *Problem description*

This problem was introduced in [Wall and Ramm, 1998], and rapidly gained popularity, subsequently it was also studied in [Hübner et al., 2004] and used by [Steindorf, 2002] to validate their fluid-structure solution algorithms. In [Dettmer and Perić, 2006, Perić et al., 2006, Dettmer and Perić, 2007], among all the fluid-structure interaction cases studied numerically, some results concerning this example are given. It can be considered as one of the standard benchmark for fluid-structure interaction. Some modifications are proposed in [Turek and Hron, 2006], in order to link this study case to the fluid benchmark proposed in [Schäfer and Turek, 1996] and used in Sec. 1.2.4 to validate the FV approach, but these are not used herein.

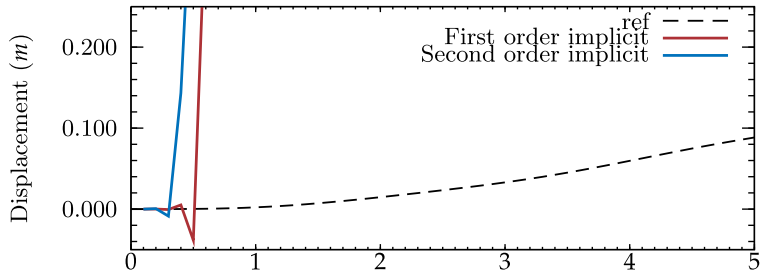
A fixed square bluff body, with a flexible appendix attached to it, is immersed in an incompressible flow (see Fig. 4.13) filling the whole domain. At a sufficiently long distance from this body, the flow is uniform with an imposed velocity $\overline{v}$ so that the Reynolds number with respects to the characteristic size of the obstacle is 330. For such a Reynolds number, the flow exhibits a transient behavior with vortices separating from the corner of the square and leading to the well-known von Kármán street. The vortices induce alternative drop and increase in the pressure field behind the rigid bluff body at a frequency that depends on the Reynolds number and the shape of the bluff body [Roshko, 1952]. The vortex shedding induces oscillations of the flexible appendix.



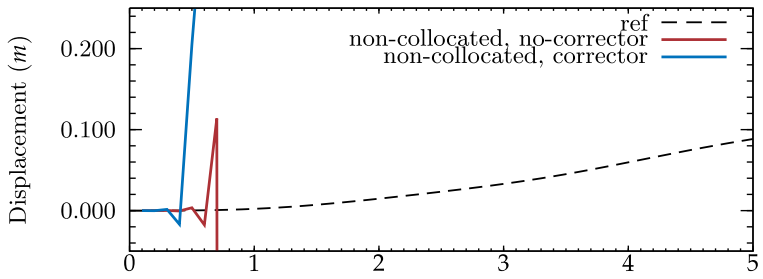
(a) Explicit coupling: influence of the predictor order



(b) Explicit coupling: influence of the window (coupling time step) size



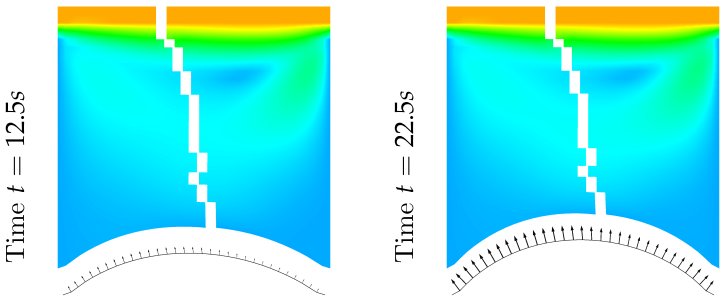
(c) Explicit coupling: influence of the fluid time integration



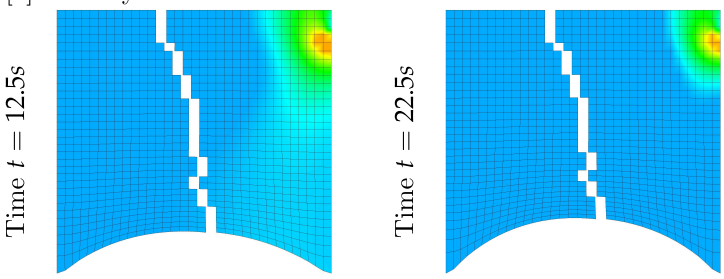
(d) Explicit coupling: influence of non-collocated schemes

Figure 4.11: Divergence of the explicit coupling for different coupling algorithm and time integration schemes

The thickness of the beam as well as the material properties are chosen so that its first eigen-frequency is close to the frequency of the vortex shedding. The



[a] Velocity field and structure with interface forces



[b] Pressure field and fluid deformed mesh

Figure 4.12: Lid driven cavity with fluid computation parallelized on two processors. The fluid domain is split by METIS.

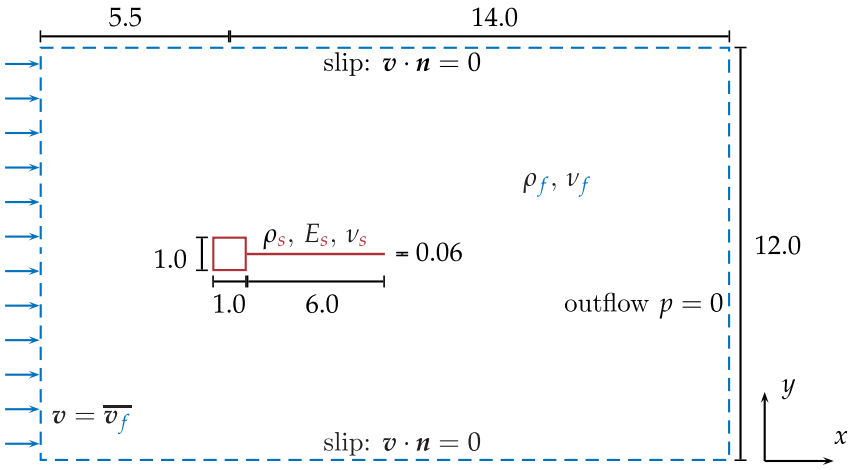


Figure 4.13: The benchmark used for FSI problems

material parameters are respectively $\rho_f = 1.18 \times 10^{-3} kg.m^{-3}$ for the density and $\nu_f = 1.54 \times 10^{-1} m^2.s^{-1}$ for the for the fluid (air at 20°C). The imposed velocity of $\overline{v} = (51.3 m.s^{-1}, 0)$ at the left hand side yields a Reynolds number of $\mathcal{R}e = 330$.

For the solid part, consider a density $\rho_s = 0.1 kg.m^{-3}$, Young modulus is of $E_s = 2.5 \times 10^6 N.m^{-2}$ and the Poisson ratio coefficient of $\nu_s = 0.35$. The

first eigen-frequency of the problem $f = 3.03s^{-1}$ obtained considering a linear material is closed to the natural frequency of vortex shedding behind a square bluff body at Reynolds $\mathcal{R}e = 330$.

The fluid discretization contains 5080 cells (i.e. around 20×10^3 d-o-f), that is perfectly sufficient to get an accurate representation of the flow, and therefore of the fluid loading on the structure. Thereby, in [Dettmer and Perić, 2006], 4300 elements are shown to give a converged representation of the fluid flow. The time step size for the time integration by a Euler implicit scheme is $\Delta t = 0.004s$ A PISO algorithm solves the FV discretized Navier-Stokes equation. The solvers used are PCG for pressure correction step and mesh motion equations and PBiCG for the momentum predictor.

The structure part is discretized with 20 nine-node elements, with polynomial description of large structural displacement. Neo-Hookean and Saint-Venant–Kirschoff materials are used. The time discretization is carried out by a Generalized-α scheme with the following parameters:

$$\rho_\infty = \frac{1}{2}; \beta = \frac{4}{9}; \gamma = \frac{5}{3}; \quad \text{and} \quad \alpha = \frac{2}{3}.$$

At each iteration, the linear system is solved by a direct generic solver for asymmetric matrices with real values.

4.3.2 *Implicit coupling and reference solution*

The coupling is insured with a DFMT-BGS solver. The tolerance on the interface displacement residual is set to:

$$\|\boldsymbol{r}_N^{(k)}\|_2 \leq 1 \times 10^{-7}$$

The Aitken relaxation technique is used with an initial value of 0.5. The convergence is very rapid (no more than 4 iterations are required to reach the required tolerance) and the relaxation parameter rapidly increases to 1. The characteristic results for some time steps are given in Fig. 4.14 in terms of velocity and pressure field. The deformation of the structure reveals the oscillations dominated by the first mode.

The displacement at the free-end of the structure is plotted in Fig. 4.15 for both Saint-Venant–Kirschoff and Neo-Hookean solid materials. The two results are very close, since the deformation remain sufficiently small. The long term response (see Fig. 4.16) indicates an almost harmonic response dominated by the first eigen-frequency of the structure.

In Fig. 4.16 the results obtained are also compared to the maximum amplitude of motion obtained from other works. Despite a well-known sensitivity of the computed result with respect to the initial condition [Hübner et al., 2004], we get the answers obtained very close to those many results from the literature obtained either by a monolithic approach or a partitioned approach [Wall, 1999] and [Dettmer and Perić, 2007, Steindorf, 2002] which are both based upon a FE discretization for both fluid and solid parts.

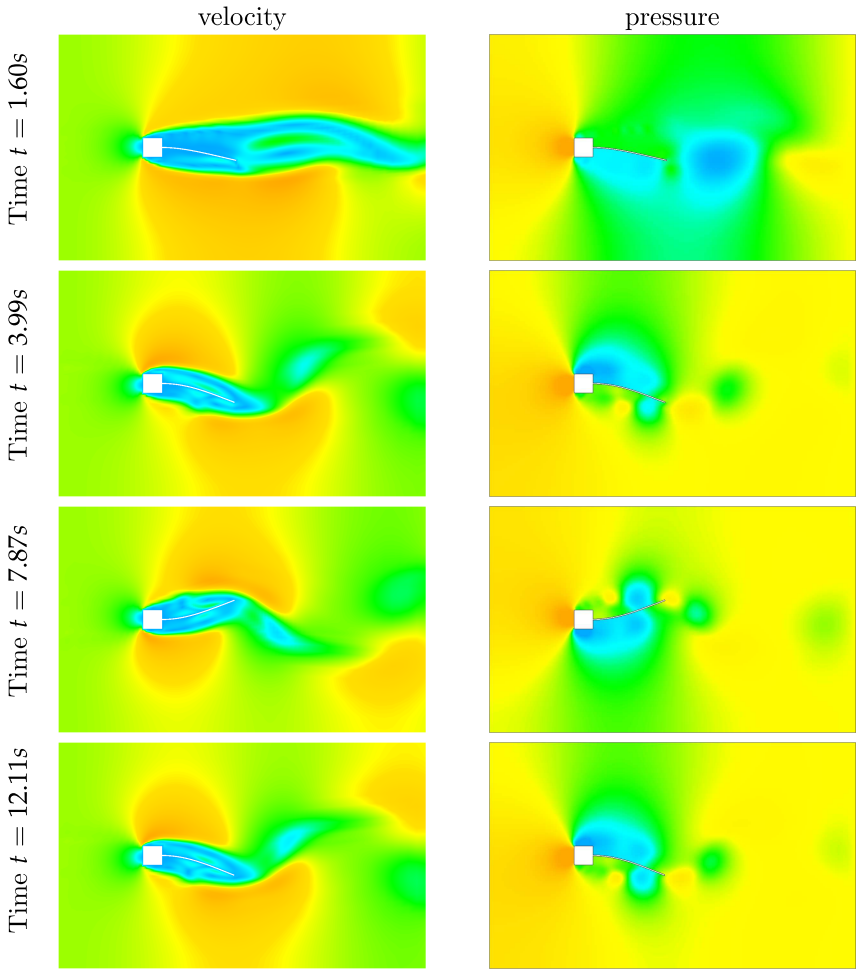


Figure 4.14: Oscillating appendix in flow: velocity and pressure field snapshots

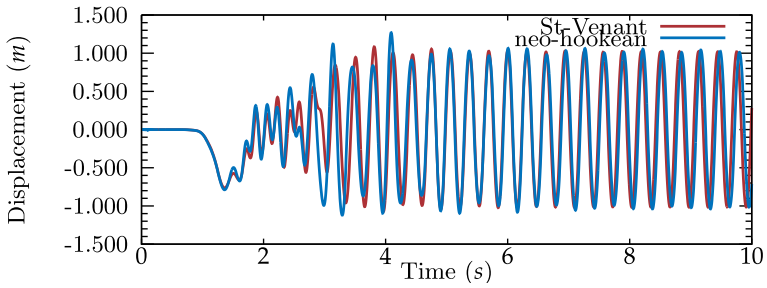


Figure 4.15: Implicit coupling: displacement of the appendix extremity for two non-linear materials (Saint-Venant and Neo-Hookean)

4.3.3 *Explicit coupling*

Contrary to the lid-driven cavity with a flexible bottom, the small number of iterations required to solve the fluid-structure interaction problem of the os-

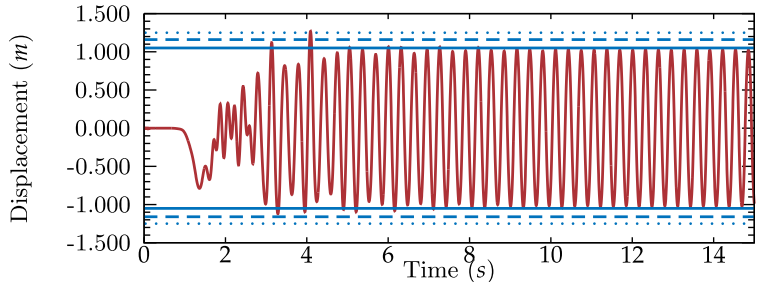


Figure 4.16: Implicit coupling: long term response with implicit coupling. Amplitude comparison with [Dettmer and Perić, 2007] (dotted line), [Steindorf, 2002] (solid line), [Wall, 1999] (dashed line).

cillating appendix suggests that an explicit coupling is capable of solving this problem. The results from explicit DFMT algorithm presented in Fig. 4.17 for the free-end displacement compare to a reference solution obtained with an implicit computation.

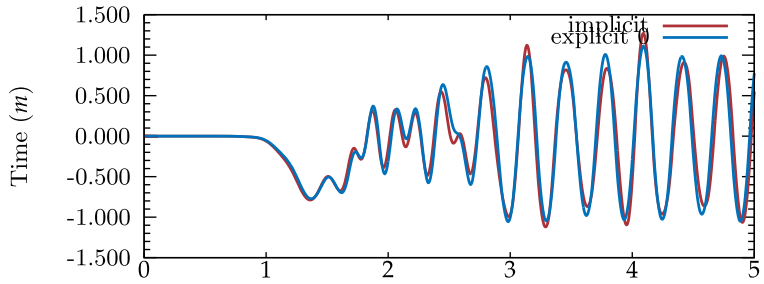
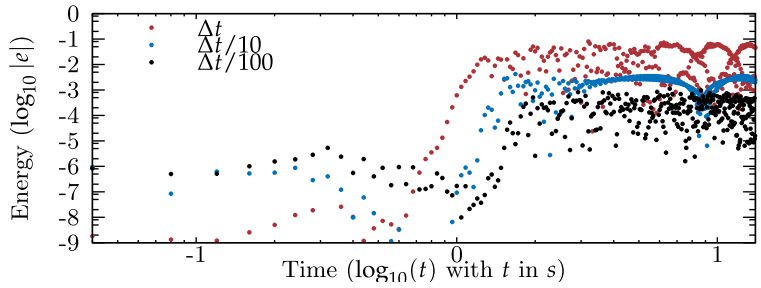


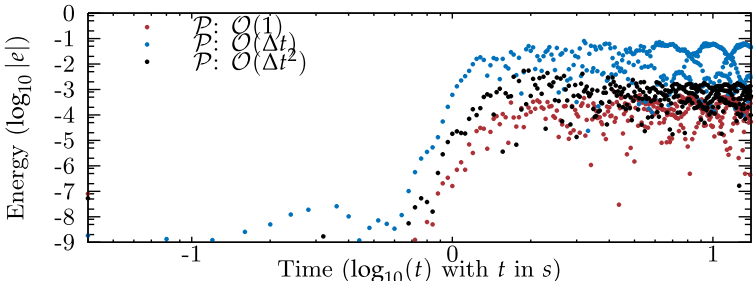
Figure 4.17: Comparison between explicit and implicit computations for the displacement of the appendix.

Using better predictors is supposed to reduce the errors made in term of residual and energy. In Fig. 4.18(b), the energy error function depending on time is represented for the zero, first and second order predictor. All the results confirm the trends we expected, with a decrease of errors when the predictor order increases.

The final study is then considered with respect to the size of time steps. In Fig 4.19, the maximum residual error on the time interval $t \in [0, 15s]$ is presented as a function of the time step size. The error is observed to decrease with a decreasing time step size. However, when the time steps become too small, the added mass effect triggers the divergence of the computation. Thus, only the less sensitive schemes with a zero order predictor are able to solve the coupled problem with the smallest time step.



(a) Energy error for different time step size



(b) Energy error for different predictors

Figure 4.18: Energy error at the interface for different explicit coupling schemes.

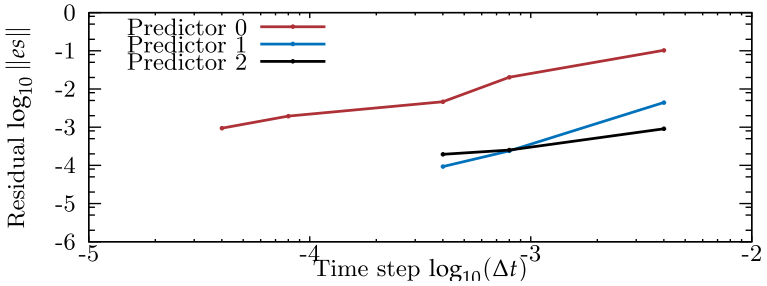


Figure 4.19: Maximum residual error for explicit coupling schemes with different time step sizes and predictors.

4.4 Three-dimensional flag in the wind

4.4.1 *Problem description*

This problem was recently introduced in [von Scheven, 2009], as a 3D generalization of the oscillating appendix problem discussed in the previous section. It can be thought of as a simplified three-dimensional model for the interaction of a flag with an incompressible viscous flow. All dimensions and the geometry of the problem, defined in Fig. 4.20, are given in *cm*.

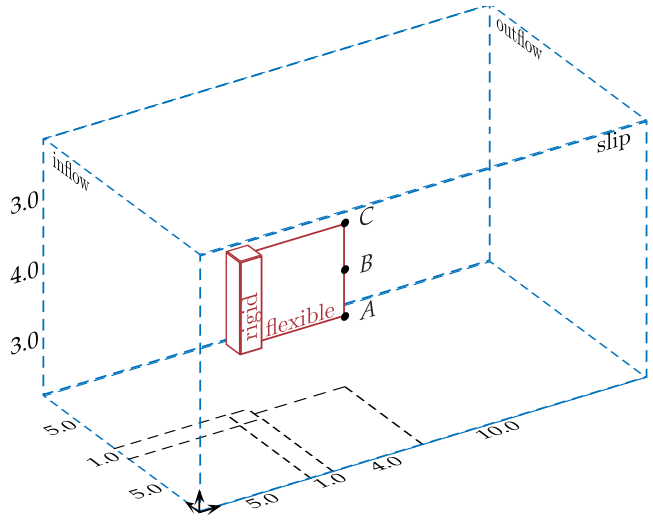
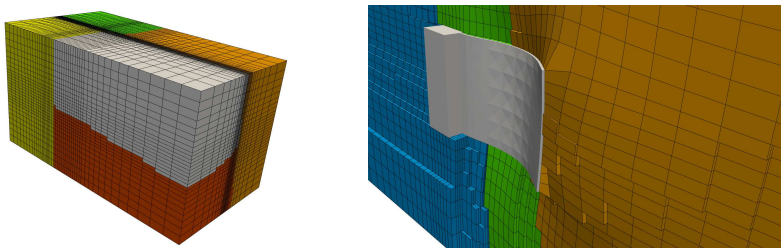


Figure 4.20: Flag in the wind: geometry and boundary conditions



(a) Fluid domain decomposed

(b) Zoom on the deformed mesh

Figure 4.21: Flag in the wind: Decomposed fluid domain and zoom on the structure

4.4.2 *Fluid discretization*

The fluid problem is discretized with FV method, and then split in 6 subdomains by METIS in order to performs parallel runs (see Fig. 4.21). The material properties are taken from [von Scheven, 2009]: the mass density $\rho_f = 1.18 \times 10^{-3} kg.cm^{-3}$ and fluid kinematic viscosity $\nu_f = 0.1542 cm.s^{-2}$.

The boundary conditions are specified in Fig. 4.20. For lateral walls of the fluid domains, the velocity boundary condition allows the slipping. At the inflow, a constant velocity is imposed with $\overline{v} = (100 cm.s^{-1}, 0, 0)$

In order to smooth the first steps, the fluid-structure interaction computations is not started from the rest, but rather follow the fluid only computation is carried out from $t = -2s$ to $t = 0s$, with the inflow velocity increase in a smooth way the velocity defined as:

$$\overline{v} \cdot \boldsymbol{e}_x = \frac{1}{2}\left(\sin\left(\pi\left(\frac{t+1}{2}\right)\right)+1\right) v_0 \qquad \text{with } v_0 = 100 cm.s^{-1}$$

The structure motion and fluid-structure interaction wall start at $t = 0s$. All the points of the fluid mesh will move in the ALE strategy, with their motion governed by a smoothing process based on a Laplacian operator and a diffusivity coefficient whose value depends on the distance to the flag. The fluid discretization techniques and solvers are equivalent to the one used in the two-dimensional example described in Sec. 4.3.

4.4.3 *Solid discretization*

In the original problem [von Scheven, 2009], the discretization of the solid problem is performed by shell finite elements. Herein three-dimensional elements with quadratic shape functions are used, with each elements therefore containing 27 nodes. Two mesh grading are used: the coarse mesh with 663 nodes, and the fine mesh with 2475 nodes. The material properties used for the solid are: a neo-Hookean elastic material with Young modulus $E_s = 2 \times 10^6 Pa$ and Poisson's coefficient $\nu_s = 0.35$ and a density $\rho_s = 2.0 kg \cdot m^{-3}$. The model can undergo finite deformation.

The time integration is handled by a generalized-α scheme with the same parameters as the one used for the two-dimensional case. At each iteration, the linearized system of solid equations of motion is solved by a direct solver for real value asymmetric matrices.

4.4.4 *Coupling*

The total number of d-o-f for the coupled problem is recalled in Table 4.2.

	fluid		solid		time
Discretization	cells	d-o-f	nodes	d-o-f	steps
Coarse	37×10^3	149×10^3	663	1989	6×10^3
Fine	290×10^3	1159×10^3	2475	7425	6×10^3

Table 4.2: Number of d-o-f for coarse and fine discretization of the three-dimensional oscillating appendix

The total number of d-o-f for the coupled problem are recalled in Table 4.2. The computation is run with a coupling time step of $1 \times 10^{-3} s$ for both the coarse and the fine mesh. The coupling scheme used is DFMT-BGS with Aitken's relaxation. The initial parameter is $\omega = 1.0$. The absolute tolerance considered is:

$$\|\boldsymbol{r}_N^{(k)}\|_2 \leq 1 \times 10^{-7}$$

As for the two-dimensional example, it should have been possible to consider explicit coupling, since the convergence of implicit scheme with the predictor of second order is fast and it does not require more than 4 to 5 iterations per time step. But in this work, only results obtained with implicit DFMT-BGS algorithm are given.

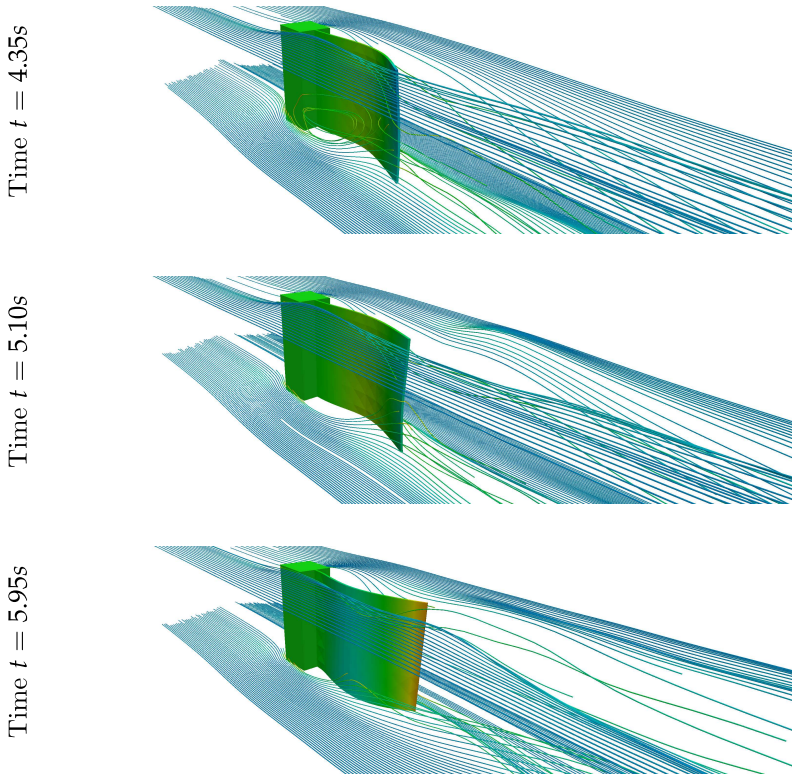


Figure 4.22: Flag in the wind: motion of the structure and stream-tube snapshots for some time steps.

4.4.5 *Results*

It is rather difficult to select the most pertinent results for three-dimensional flow problems, and even more for coupled problems. For start, in order to obtain a qualitative picture Fig 4.23 represents the stream-tubes going through the two lines $(x = 6.0, y, z = 3.0)$ and $(x = 6.0, y, z = 7.0)$ along with the deformed shape of the flag.

The displacement of three points at the free-end represented in Fig. 4.23, show that the motion of the flag corresponds to the first flexural mode. It is interesting to note that the results in [von Scheven, 2009] indicate that, after a certain time, some torsion modes occur on the flag, that could not be confirmed here at the same Reynolds number. Consequently, the motion amplitude presented herein is around 4 times bigger than the one obtained in [von Scheven, 2009]. Our results are more in agreement with the ones provided for 2D version of the same problem (see [Wall, 1999]) with a flexible beam. It is hard to predict the exact result and to say *a priori* which results are closer to the exact solution, for such a complex flow in three-dimensions with a relatively high Reynolds number.

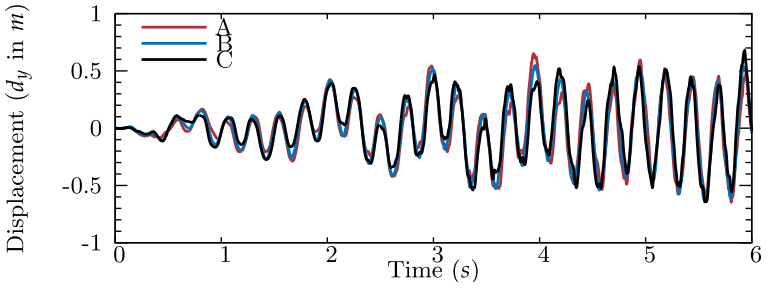


Figure 4.23: Oscillating three-dimensional flag: extremity displacements for points $A = (10.0, 5.5, 3.0)$, $B = (10.0, 5.5, 5.0)$ and $C = (10.0, 5.5, 7.0)$.

4.5 Two-dimensional sloshing wave hitting a flexible structure

4.5.1 *Introduction and a first approach*

In [Kassiotis et al., 2008], a partitioned strategy based on a three-software coupling is proposed to model the propagation of a tsunami wave and its impact on a submerged structure. For this approach, the coupling between a Boundary Element Method (BEM) based software solving the non-linear equations used to model a free-surface flow [Srisupattarawanit et al., 2006]. The flow is supposed to be inviscid, and the solve the linear Laplace equation is only required inside the fluid domain while the kinematics and dynamic boundary conditions at the free-surface are fully non-linear [Dias and Dutykh, 2006, Fochesato et al., 2007]. This approach is able to represent flow propagation at a large scale, but fail to represent the flow near the structure accurately. For that reason, near the submerged structure, the Navier-Stokes equation is used. However the representation of the free-surface flow is not able to handle complex flow such as the breaking of the wave.

For such a problem, a promising approach is based on mesh-less methods [Fries, 2005] that are able to represent any domain deformation. The SPH is for instance used to achieve that aim. However, this approach requires to take special care at the fluid-structure interaction interface and to handle the transition between mesh-less and mesh-based methods with caution. In [Idelsohn et al., 2003], another approach based on fast re-meshing of a Lagrangian representation of a flow discretized by FEM is proposed to deal with the wide deformation of the free-surface fluid domain.

Use of a multiphase representation of the fluid domain to model the free-surface flow is chosen herein (See Sec 1.3.2.2 for a short description of the method). The drawback of such an approach is that it requires to model the two sides of the free-surface flow (here water and air). However, the meshing is simplified, and a Euler approach can be used when the domain is not moving. For the fluid-structure interaction, an ALE approach generalizes the previous approach.

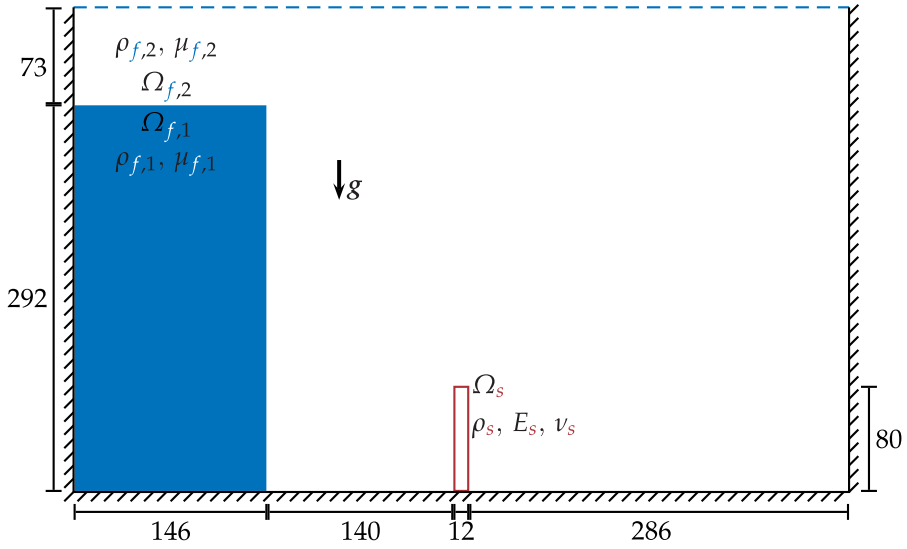


Figure 4.24: Dam break interacting with an obstacle: geometry and boundary conditions

4.5.2 *Problem description*

The problem solved is a modification of the dam-break problem presented in Sec. 1.3.2.2. At initial time $t = 0s$, the same water column starts to fall under the gravity loading. Instead of hitting a rigid structure, its geometry is modified in order to get a more slender obstacle (Fig. 4.24). Furthermore, the material properties of the obstacle are modified in order to obtain a flexible elastic structure. This problem was studied experimentally and numerically [Walhorn et al., 2005] by a monolithic approach based upon discontinuous Galerkin FE [Walhorn, 2002, Hübner et al., 2004].

4.5.3 *Fluid discretization*

At all boundaries a non-slip boundary condition is applied, except for the upper boundary $z = H$ where the tank is open and a total pressure condition is used:

$$p + \frac{1}{2}\rho\|\boldsymbol{v}\|^2 = p_{\text{atm}} \tag{4.1}$$

with $p_{\text{atm}} = 0$.

The material properties are imposed as follows: for the high density fluid the density and the kinematic viscosity are $\rho_{f,1} = 1 \times 10^3 kg.m^{-3}$ and $\nu_{f,1} = 1 \times 10^6 m.s^{-1}$, whereas $\rho_{f,2} = 1 kg.m^{-3}$ and $\nu_{f,2} = 1 \times 10^5 m.s^{-1}$ are considered for the low density domain. Due to the scale of the simulation no surface tension need to be considered.

Mesh motions based on smoothing operator like the Laplacian equation fails for the used meshes since the points around the obstacle have difficulties to

follow the large displacement and rotations of the mesh around the structure. For this reason, the mesh motion problem is solved by using a pseudo-elastic model where the stiffness is a quadratic inverse function of the distance to the interface between solid and fluid.

The fluid problem is discretized with Finite Volume method. The results for two meshes with respectively 3340 and 13760 cells are presented. The fluid is handled by second order space discretization and a Van Leer limiter is used for the advected terms. The time integration scheme is implicit Euler. The fluid domain was not split and parallelized, but acceleration of the computational time was obtained using a Generalized Algebraic-MultiGrid (GAMG) linear solver.

Small time steps are required by the explicit nature of the coupling between the phase function indicator problem, the momentum prediction and the pressure correction step.

4.5.4 *Structure discretization*

For this problem, it is proposed herein to use two dimensional elements with quadratic shape functions. Each element therefore contains 9 nodes. For the coarse grid, 51 nodes are considered while 165 nodes are used to discretize the fine grid. The material properties used for the solid are: a neo-Hookean elastic material with Young's modulus $E_s = 1 \times 10^6 Pa$ and Poisson's coefficient $\nu_s = 0$ and density $\rho_s = 2500 kg \cdot m^{-3}$.

The model can endure finite deformation and the time integration is handled by a Generalized-α scheme with:

$$\rho_\infty = \frac{1}{2}; \beta = \frac{4}{9}; \gamma = \frac{5}{3} \quad \text{and} \quad \alpha = \frac{2}{3} \tag{4.2}$$

At each iteration, the linear system is solved by a direct solver for real value asymmetric matrices.

4.5.5 *Coupling*

	fluid		solid		time
Discretization	cells	d-o-f	nodes	d-o-f	steps
Coarse	3440	17.2×10^3	51	102	1×10^4
Fine	13760	68.8×10^3	165	330	5×10^4

Table 4.3: Number of d-o-f for coarse and fine discretization of the two-dimensional dam-break problem

The computation (with total number of d-o-f given in Tab. 4.4) is run with a time step of 1×10^{-4} for the coarse discretization and 2×10^{-5} for the fine one. The coupling scheme used here is DFMT-BGS with Aitken's relaxation, and the initial parameter is $\omega = 0.25$. The predictor is of order 1, since computations with second order predictor fails for the finest grid. The absolute tolerance considered is:

$$\|\boldsymbol{r}_N^{(k)}\| \leq 1 \times 10^{-6}$$

In Fig. 4.25 the number of iteration required to reach the convergence criteria is given. Note that their is no iteration required before the water hits the structure (the effect of air flow can almost be deemed as negligible for this structure). Then, the number of iteration depends on the discretization density. After the breaking of the dam, the finest scale is able to represent fine water slosh that affects the convergence of the implicit coupling.

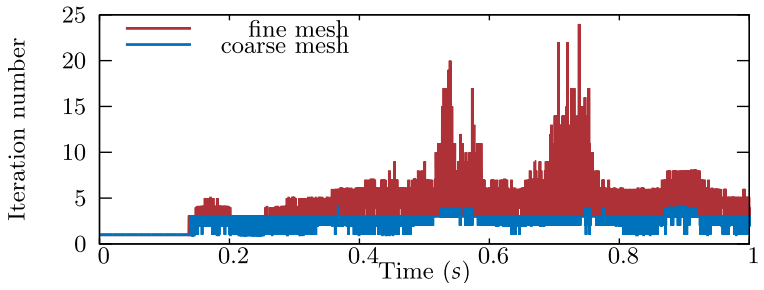


Figure 4.25: Number of iteration in order to make the DFMT-BGS algorithm converge for the two-dimensionnal dam-break example

4.5.6 *Results*

In Fig. 4.27, the high density fluid domain is represented, as well as the streamlines for the high and low density fluid domains. During the first $0.1s$ of the simulations, the water column falls under the gravity loading. There is no effect whatsoever on the structure until the high density fluid reaches it. The maximum amplitude of the motion is obtained at $t = 0.25s$, before the solid comes back to its initial position due to flow friction.

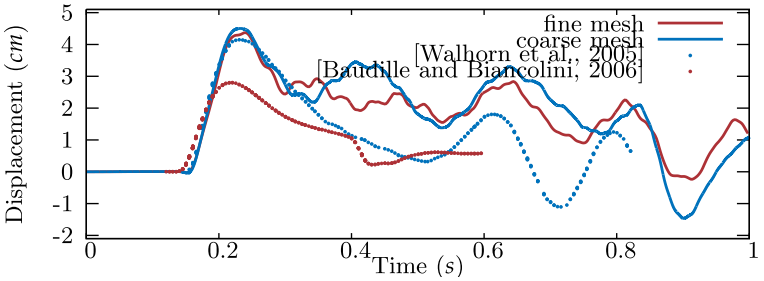


Figure 4.26: Obstacle extremity for different meshes and comparison with results from literature

After one second of simulation, the fluid is not entirely at rest, but its main effects on the structure were captured. Note that before the flow goes back from hitting the right-hand wall, all the numerical results are somehow in agreement. After, the impact high density fluid is highly fragmented (see Fig. 4.27 for time greater than $0.3s$), and the results depend on the ability of the fluid part to represent these fine effects. However, one can notice, the main effect on the structure is when the wave first hits the structure, and this is captured well.

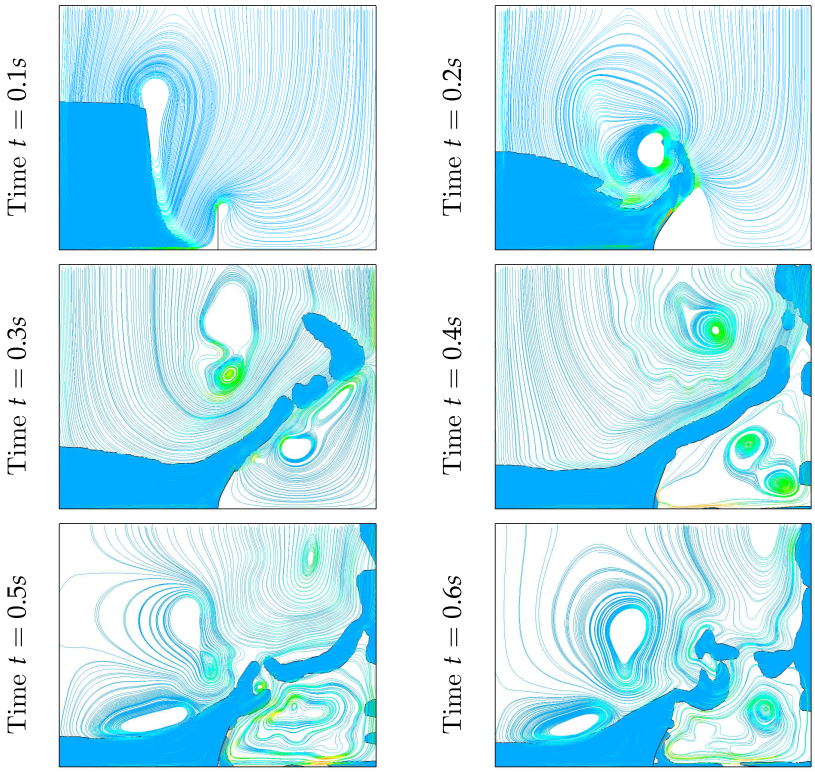


Figure 4.27: Bi-dimensional dam break problem. Evolution of the free surface, structure motion and streamlines for water and air.

4.6 Three-dimensional sloshing wave impacting a flexible structure

4.6.1 *Problem description*

The problem solved in this example is a simplified representation of the dam-breaking event that brings about a sloshing wave impact on a flexible structure presented in Fig. 4.28. At initial time $t = 0s$, a three-dimensional water column starts to fall down under the gravity loading and eventually hits the obstacle placed in the way. The obstacle is a slender plate-like body made of elastic material that can undergo large deformation.

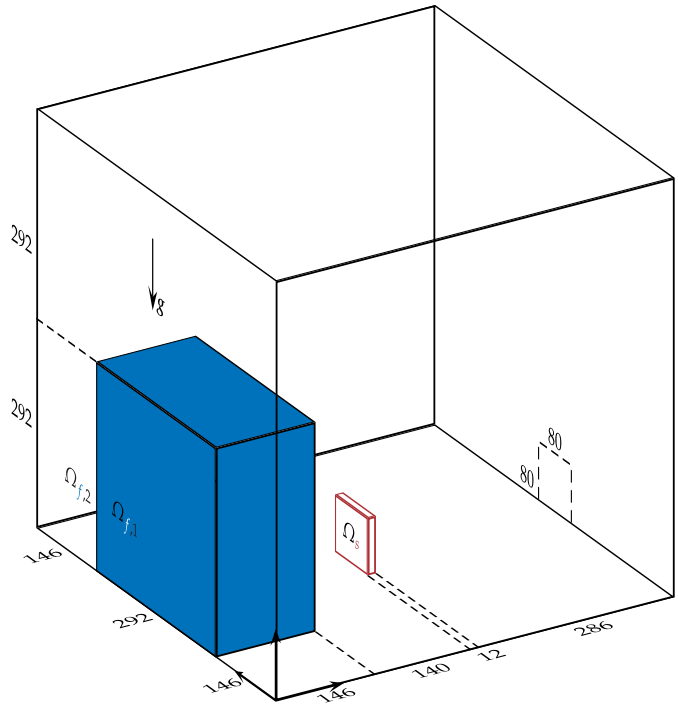


Figure 4.28: Three-dimensional wave impacting an obstacle: geometry and boundary conditions

The dimensions of the problem are given in Fig. 4.28 as well as the boundary conditions.

4.6.2 *Fluid discretization*

We do not want that the water bounce-back and again hit the structure after breaking on the walls. For that reason, only the left and bottom planes of the fluid domain are defined as non-slipping walls, while the others are defined with atmosphere boundary condition for the pressure as defined in Eq. (4.1).

The material properties are chosen as follows: for the high density fluid (water in the reservoir) the density and the kinematic viscosity are $\rho_{f,1} = 1 \times 10^3 kg.m^{-3}$ and $\nu_{f,1} = 1 \times 10^6 m.s^{-1}$, whereas for the low density fluid (air in the remaining part of the domain) $\rho_{f,2} = 1 kg.m^{-3}$ and $\nu_{f,2} = 1 \times 10^5 m.s^{-1}$. The mesh motion problem is solved using a Laplacian smoothing in which the diffusion coefficient is a quadratic inverse function of the distance to the interface between solid and fluid.

The fluid problem is discretized with Finite Volume cells. The results are computed for two meshes with the chosen discretization and the number of cells given in Tab. 4.4. The fluid is handled by second order space discretization and a Van Leer limiter is used for the advection terms. The time integration scheme is implicit Euler. For such a scale of modelling it is not required to consider surface tension between the two fluids. For this problem the fluid domain was not split and parallelized, but reduction of the computational time was obtained by using a Generalized Algebraic-MultiGrid (GAMG) linear solver.

Note that small time steps are required for the explicit solution of the phase function indicator equation, as well as the half-implicit nature of the coupling between the momentum predictor and the pressure corrector.

4.6.3 *Solid discretization*

We propose here to use three-dimensional elements with quadratic shape functions, where each element has 27 nodes. The material properties used for the solid are: a neo-Hookean elastic material with Young modulus $E_s = 1 \times 10^6 Pa$ and Poisson's coefficient $\nu_s = 0$ and a density $\rho_s = 2500 kg \cdot m^{-3}$, which can represent finite deformation. The time integration is carried out by a Generalized-α scheme with the same parameters as the one used for the previous two-dimensional dam-break case (see Sec. 4.5).

4.6.4 *Coupling*

	fluid		solid		time
Discretization	cells	d-o-f	nodes	d-o-f	steps
Coarse	13×10^3	63×10^3	363	1.1×10^3	1×10^5
Fine	104×10^3	520×10^3	2205	6.6×10^3	1×10^5

Table 4.4: Number of d-o-f for coarse and fine discretization of the three-dimensional dam-breaking problem

The computation of the coupled problem (with total number of d-o-f given in Tab. 4.4), is carried out by an implicit iterative scheme. The results of fluid and solid computations are matched for a time step of 1×10^{-4} for the coarse discretization and 2×10^{-5} for the fine one. The coupling scheme used is DFMT-BGS with Aitken's relaxation. The initial parameter is $\omega = 0.25$ and the predictor is of order 1. The absolute tolerance considered is:

$$\|\boldsymbol{r}_N^{(k)}\| \leq 1 \times 10^{-6} \tag{4.3}$$

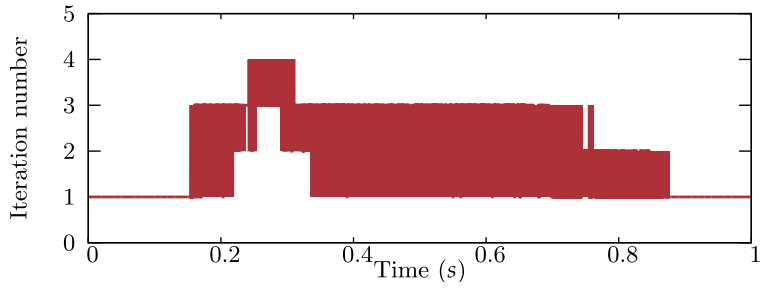


Figure 4.29: Number of iterations in order to make the DFMT-BGS algorithm converge for the three-dimensional dam-breaking problem

In Fig. 4.29 the number of iterations required to reach the convergence criteria is given. Note that there is no iteration required before the water hits the structure (the effect of air flow can almost be deemed negligible with respect to the structure). Then, the number of iteration depends on the discretization density. In reaching the opposite wall, the water does not rebound on the wall but simply flows away.

4.6.5 *Results*

In Fig. 4.31, the high density fluid domain is represented, as well as some part of the fluid mesh and the structure displacement. The first $0.1s$ of the simulations, the water column falls under the gravity loading. There is no effect whatsoever on the structure until the high density flow reaches its bottom. The maximum amplitude of the motion is obtain at $t = 0.25s$, before the solid comes back to its initial position and oscillates after the shock.

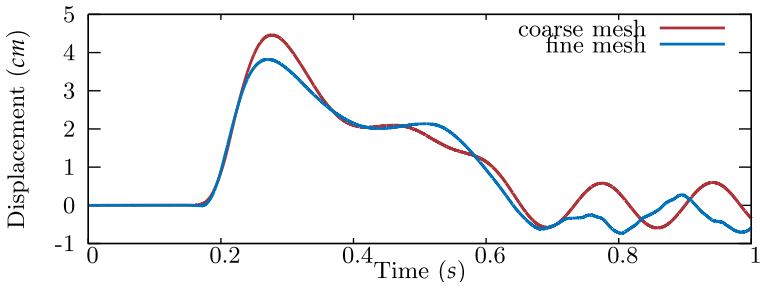


Figure 4.30: Three-dimensional dam break example: obstacle extremity displacement

In Fig. 4.30 the motion of the top of the obstacle is plotted. Contrary of the two-dimensional example, smaller drops of high density fluid are not interacting with the obstacle after the main interaction. Therefore, the motion of the solid part is rather well described with the coarsest grid.

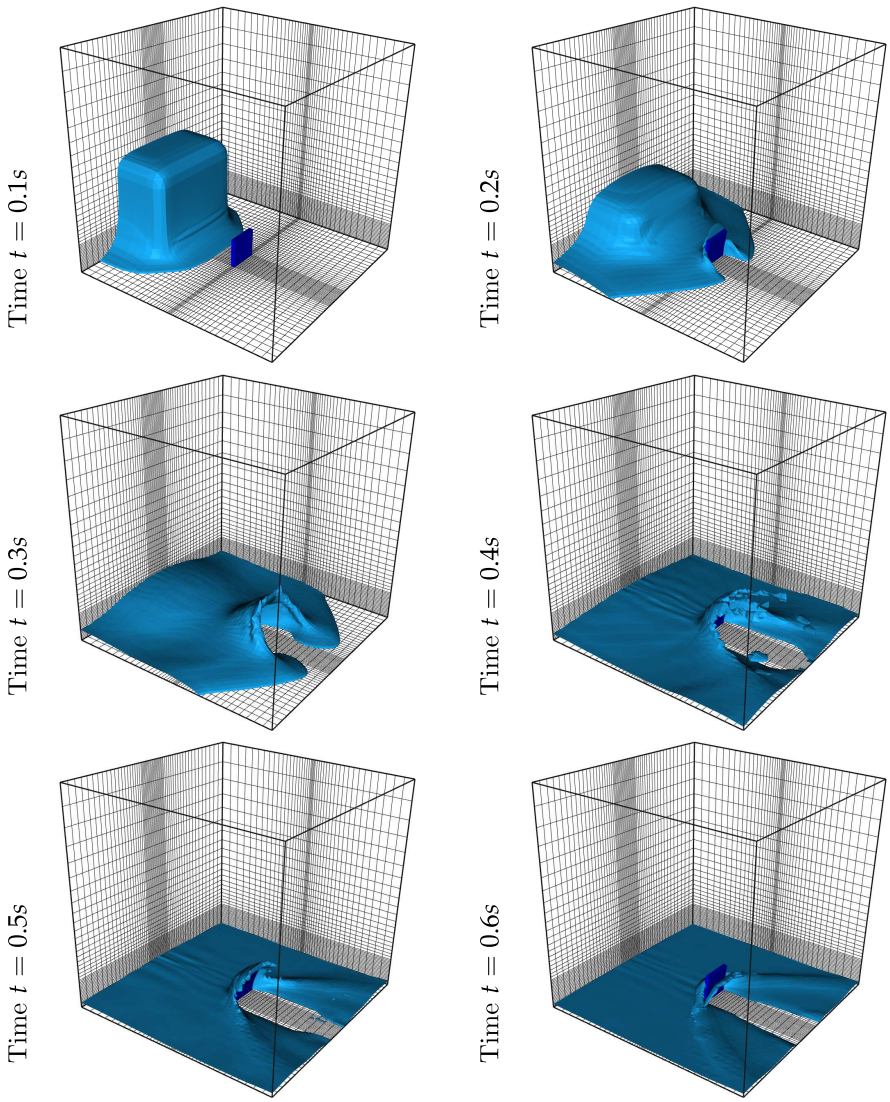


Figure 4.31: Tri-dimensional dam break problem. Evolution of the free surface and motion of the structure.

Closure

The main goal of this chapter was to present the capacities of the presented partitioned framework for fluid-structure interactionproblems. For some cases it is possible to compare the results obtained with those obtained by other authors. All the results presented hereby are within a comparable order of magnitude.

The added-mass effect does not allow explicit computation for some geometries, flows and mass ratio-between solid and fluid. In this cases, iterations with DMFT-BGS need to be performed. Aitken's relaxation is required to insure the convergence or accelerate the iterative process. In order to improve error made at the initial step, the use of predictors is possible. When explicit coupling is sufficient, the errors made at the interface depends on the size of the time step as well as the order of the predictor.

The middleware CTL and the coupling of existing codes allow good performances, and therefore, problems with large numbers of d-o-f can be reached (more than a billion on the fluid side). Concerning fluids, the coupling with free-surface flow is also explored. The three-dimensional applications in this context is an original application of the proposed strategy.

Conclusion

The present dissertation deals with a partitioned strategy for multiphysics problems specifically applied to the fluid-structure interaction. More precisely this strategy is part of the Direct Force-Motion Transfers algorithm framework between two sub-problems, solved implicitly by a fixed-point iterative strategy and combined with a dynamic relaxation parameter in order to accelerate convergence. The algorithm used is known to be conditionally stable; the corresponding criterion of the proposed DFMT-BGS algorithm in the context of fully nonlinear analysis is hereby demonstrated (see App. A). In particular, the rigorous proof is given for a coupling algorithm where the data exchanges between iterations are primal (and not dual, or in terms of Lagrangian multipliers).

The method allows to perform coupled simulations with existing codes that were not initially developed to support a multiphysics approach. The component technology presented in this work is used to allow the coupling between software products. The communication between the components is insured by the CTL [Niekamp, 2005b], where the use of RPC and no file exchange, are found to give excellent performances. Furthermore, an important cost of CFD subproblem make it interesting to employ parallel computation for the fluid component itself. Therefore, for the first time, a CTL component was built herein using the parallel features of an existing engineering software. The overall performances of the programmed component allow to solve problems with a large number of d-o-f, many time steps, fully three-dimensional coupled problems, as well as further extending the solution capabilities with two level parallelization.

The coupled problems solved herein utilize very different numerical strategies: FE for the solid part, FV for the fluid part. The use of these popular methods for the fluid and solid parts allows to benefit from the advanced features of the two families of methods, each developed by the experts from the corresponding domain. Accordingly, on the fluid side, it is possible to use a very efficients semi-implicit solver for incompressible flow (PISO), solution techniques (Algebraic Multigrid) or advanced flow models. For instance, it is referred to the interaction between a structure and a sloshing wave where free-surface flow is modeled by a two-phase solver.

Several development presented in this work can further be improved. The current implementation explores the DFMT-BGS solver, only one among implicit coupling solvers that are known to be conditionally stable. All the numerical simulations presented in this dissertation concern the density ratios between the structure and the fluid parts that are sufficient to ensure the convergence of

the computations in the spirit of the presented proof. The need to implement a more elaborate iterative schemes for coupled problems such as quasi-Newton strategy might be necessary to tackle light-weight structures interacting with flows. Furthermore, the interpolation between the fluid and the structure parts is handled by a meshless method that is quite expensive ($\mathcal{O}(N^3)$ with N the number of nodes at the interface). Improving the interpolation using a basis of compact support radial functions could be an interesting first step.

The perspectives broadened by this work are numerous. It would be challenging to use turbulence models in order to represent more realistic wind flows interacting with structures. Another interesting point would be to simulate the whole process of a tsunamis wave hitting a structure (from its generation to its propagation). In this case, the importance of the free-surface representation problem would require three-software coupling, with one of them a BEM code able to represent non-linear shallow water equations briefly explored in [Kassiotis et al., 2008].

From the structure point of view, the use of more advanced models would also be of interest. For instance in [Hautefeuille, 2009], a multiscale framework is proposed to model the mechanical behavior of concrete from its micro-scale. For other physical properties (*e.g.* porosity, thermal conductivity...) the interaction with liquid, which is known to be crucial for process such as cement hydration, concrete drying and other phenomenon typical of its behavior at young age. Therefore it would be interesting to develop a multiscale/multiphysics models both for solid and fluid by using the tools presented in this thesis.

A

Stability and convergence of the Direct Force-Motion Transfert Block-Gauss-Seidel algorithm

A.1 Reformulation of the fluid-structure interaction problem in a Differential Algebraic Equation framework

The main goal of this section is to confirm the stability of the partitioned approach for coupled fluid-structure interaction problems by showing a stable error propagation of the DFMT-BGS algorithm. To that end, it is assumed that the most suitable choice of integration schemes is made to solve each partition and that each scheme used for the fluid and the structure sub-problem is convergent and stable[1]. By further considering the DFMT-BGS algorithm, formulated in the general framework of *differential-algebraic equations* (DAE), a proof of nonlinear stability of the partitioned algorithm can be provided, following the general procedure provided in [Arnold and Gunther, 2001].

The first sub-problem related to the fluid flow on a moving domain discretized by Finite Volume (Eq. (1.33)) can be written as:

$$\begin{aligned} 0 &= r_f(\boldsymbol{x}_f(t), \boldsymbol{x}_s(t), \boldsymbol{y}_f(t), \boldsymbol{\lambda}(t)) \\ &= \begin{bmatrix} \mathbf{K}_m \mathbf{u}_m - \mathbf{D}_m \mathbf{u}_s \\ \mathbf{M}_f \dot{\mathbf{v}}_f + \mathbf{N}_f(\mathbf{v}_f - \dot{\mathbf{u}}_m)\mathbf{v} + \mathbf{K}_f \mathbf{v}_f + \mathbf{B}_f \mathsf{p}_f - \mathbf{f}_f - \mathbf{D}_f^T \boldsymbol{\lambda} \\ \mathbf{B}_f^T \mathbf{v}_f \end{bmatrix} \end{aligned} \tag{A.1}$$

where fluid velocities $\mathbf{v}_f$ and the mesh displacements $\mathbf{u}_m$ are gathered in $\boldsymbol{x}_f = (\mathbf{v}_f, \mathbf{u}_m)$, while their time derivatives as well as the pressure field are denoted $\boldsymbol{y}_f = (\dot{\mathbf{v}}_f, \mathsf{p}, \dot{\mathbf{u}}_m)$. The force at the interface are denoted with $\boldsymbol{\lambda}$ and matrix $\mathbf{D}_f$ denotes the result of their interpolation across the interface from the fluid side. The residual r_f gathers the mesh motion for ALE, the coupled discretized momentum equation and incompressibility condition. Rewriting the incompressibility condition using the acceleration in order to reduce the order of the DAE

[1] It is also assumed that each sub-problem, fluid and structure, is solved by the corresponding software product, which should to solve a fluid-structure interaction problem; The details of software coupling are discussed in Chapter 3 of this work.

associated to the fluid problem leads to:

$$\begin{aligned} 0 &= r_f(\boldsymbol{x}_f(t), \boldsymbol{x}_s(t), \boldsymbol{y}_f(t), \boldsymbol{\lambda}(t)) \\ &= \begin{bmatrix} \mathbf{K}_m \dot{\mathbf{u}}_m - \mathbf{D}_m \dot{\mathbf{u}}_s \\ \mathbf{M}_f \dot{\mathbf{v}}_f + \mathbf{N}_f(\mathbf{v}_f - \dot{\mathbf{u}}_m)\mathbf{v} + \mathbf{K}_f \mathbf{v}_f + \mathbf{B}_f \mathsf{p}_f - \mathbf{f}_f - \mathbf{D}_f^T \breve{} \\ -\mathbf{B}_f^T \mathbf{M}_f^{-1} \left(\mathbf{N}_f(\mathbf{v}_f - \dot{\mathbf{u}}_m)\mathbf{v} + \mathbf{K}_f \mathbf{v}_f + \mathbf{B}_f \mathsf{p}_f - \mathbf{f}_f - \mathbf{D}_f^T \breve{} \right) \end{bmatrix} \end{aligned} \tag{A.2}$$

Considering now the structure sub-problem, the primal variables, the solid displacements and velocities, are gathered together in the same vector $\boldsymbol{x}_s = (\mathbf{u}_s, \dot{\mathbf{u}}_s)$ and the acceleration is denoted $\boldsymbol{y}_s = \ddot{\mathbf{u}}_s$. The residual form of the equation of motion can then be written:

$$0 = r_s(\boldsymbol{x}_s(t), \boldsymbol{x}_f(t), \boldsymbol{y}_s(t), \boldsymbol{\lambda}(t)) \tag{A.3}$$

The last equation can be written as:

$$r_s(\boldsymbol{x}_s(t), \boldsymbol{x}_f(t), \boldsymbol{y}_s(t), \boldsymbol{\lambda}(t)) := \left[\mathbf{M}_s \ddot{\mathbf{u}}_s + \mathbf{f}_s^{\text{int}}(\mathbf{u}_s) - \mathbf{f}_s^{\text{ext}} - \mathbf{D}_s^T \breve{} \right] \tag{A.4}$$

where the $\mathbf{D}_s$ indicates the force distribution at the interface on the structure side.

The interface continuity equation, connecting two sub-problems can then be stated in terms of acceleration:

$$0 = r_\lambda(\boldsymbol{x}_f(t), \boldsymbol{x}_s(t), \boldsymbol{y}_f(t), \boldsymbol{y}_s(t)) := -\mathbf{D}_s \ddot{\mathbf{u}}_s + \mathbf{D}_f \dot{\mathbf{v}}_f \tag{A.5}$$

With this notation in hand, the proposed DFMT-BGS algorithm can be stated as follows: first solve the fluid sub-problem associated together with the continuity equation at the interface:

$$\begin{aligned} \partial_t \boldsymbol{x}_f{}^{(k)} &= f_{r_f}(\boldsymbol{x}_f{}^{(k)}, \boldsymbol{x}_s{}^{(k-1)}, \boldsymbol{y}_f{}^{(k)}) \\ 0 &= r_f(\boldsymbol{x}_f{}^{(k)}, \boldsymbol{x}_s{}^{(k-1)}, \boldsymbol{y}_f{}^{(k)}, \boldsymbol{\lambda}^{(k)}) \\ 0 &= r_\lambda(\boldsymbol{x}_f{}^{(k)}, \boldsymbol{x}_s{}^{(k-1)}, \boldsymbol{y}_f{}^{(k)}, \boldsymbol{y}_s{}^{(k-1)}) \end{aligned} \tag{A.6}$$

The structure sub-problem is then solved with imposed forces at the interface:

$$\begin{aligned} \partial_t \boldsymbol{x}_s{}^{(k)} &= f_{r_s}(\boldsymbol{x}_f{}^{(k)}, \boldsymbol{x}_s{}^{(k)}, \boldsymbol{y}_s{}^{(k)}) \\ 0 &= r_s(\boldsymbol{x}_f{}^{(k)}, \boldsymbol{x}_s{}^{(k)}, \boldsymbol{y}_s{}^{(k)}, \boldsymbol{\lambda}^{(k)}) \end{aligned} \tag{A.7}$$

A.2 Error propagation, stability and convergence of DFMT-BGS algorithm

Note that each nonlinear sub-problem is solved by a different solver: Newton's algorithm for the solid part, a *segregated approach* like PISO for the fluid part; the final stability results remain valid in a more general context as long as sub-problem computation remains table and convergent. Thus we only need to confirm the convergence of the iterative DFMT-BGS procedure.

The following notations are introduced: the subscript N for the restriction of the function of time t to the interval $[T_{N-1}, T_N]$, $h = T_N - T_{N-1}$, the superscripts (k) for the iteration counter of the DFMT-BGS procedure and the Δ

symbol denoting the distance of numerical approximation from the exact solution $\Delta x_N^{(k)} = x_N^{(k)} - x^\star$.

Moreover, the same time step for each sub-problem of the fluid-structure interaction are not needed, for instance the fluid problem can be solved with many small time steps on the window N. For many case, the time interpolation of the evolution has to be considered *a priori*.

The stability of DFMT-BGS operator split procedures is mainly governed by error propagation from one window N to the following $N+1$. The partitioned approach is studied in the differential-algebraic equations (DAE) framework in [Arnold and Gunther, 2001]. The cited work is applied to multi-body dynamics systems and several interesting results are given including the stability proof where the Lagrange multipliers, and not a primal variable values define at the interface are exchanged from one iteration to the next. In [Arnold, 2001], the stability criterion corresponding to the block-Gauß-Seidel algorithm such as the one used herein is given, but without any proof. These results are valid, and sucessfully applied to the DFMT-BGS algorithm for fluid-structure interaction [Matthies and Steindorf, 2002, Steindorf, 2002] used herein. In the following a detailed proof that cannot be found in the literature for the fluid-structure interaction context is given.

Same kind of results can be used for other coupled problems where the time integration schemes of the sub-problems are different, such as thermomechanics [Kassiotis et al., 2009a] or with different time scales for mechanics and thermal component and a generalized non-linear operator split for problems with internal variables [Kassiotis et al., 2009b].

Theorem 1

There exists $C \in \mathbb{R}_+$ *such that for all* $(k) > 1$

$$\max_n \left(\|\Delta \boldsymbol{x}_{f\,n}^{(k)}\| + \|\Delta \boldsymbol{x}_{s\,n}^{(k)}\| + \|\Delta \boldsymbol{y}_{f\,n}^{(k)}\| + \|\Delta \boldsymbol{y}_{s\,n}^{(k)}\| + \|\Delta \boldsymbol{\lambda}_n^{(k)}\| \right) < C \cdot \max_n \left(\mu^{k-2} \left(\|\Delta \boldsymbol{x}_{f\,n}^{(0)}\| + \|\Delta \boldsymbol{x}_{s\,n}^{(0)}\| \right) + \mu^{k-1} \|\Delta \boldsymbol{y}_{s\,n}^{(0)}\| \right)$$

where: $\mu = \alpha + \mathcal{O}(H)$ *and* α *the contraction constant characterizing the operator split procedure employed to solve the coupled problems, which is written as:*

$$\alpha = \max_n \left\| \left[\partial_{y_s} r_\lambda \left[\partial_{y_s} r_s \right]^{-1} \partial_\lambda r_s \right]^{-1} \partial_{y_f} r_\lambda \left[\partial_{y_f} r_f \right]^{-1} \partial_\lambda r_f \right\|$$

The main condition of contractive properties, $\alpha < 1$, guarantees the convergence of the operator split procedure for the given time window when $(k) \longrightarrow \infty$. We set now to apply this stability criterion to the DFMT-BGS for fluid-structure interaction presented in the previous section. We thus obtain:

i - Fluid sub-problem on a moving domain solved with FV method:

$$\partial_{y_f} r_f = \begin{bmatrix} \mathbf{K}_m & \mathbf{0} & \mathbf{0} \\ \partial_{\dot{\mathbf{u}}_m}(\mathbf{N}_f(\mathbf{v}_f - \dot{\mathbf{u}}_m)\mathbf{v}) & \mathbf{M}_f & \mathbf{B}_f \\ -\mathbf{B}_f^T \mathbf{M}_f^{-1} \partial_{\dot{\mathbf{u}}_m}(\mathbf{N}_f(\mathbf{v}_f - \dot{\mathbf{u}}_m)\mathbf{v}) & \mathbf{0} & -\mathbf{B}_f^T \mathbf{M}_f^{-1} \mathbf{B}_f \end{bmatrix} \tag{A.8}$$

and

$$\partial_{\lambda} r_f = \begin{bmatrix} \mathbf{0} \\ -\mathbf{D}_f^T \\ \mathbf{B}_f^T \mathbf{M}_f^{-1} \mathbf{D}_f^T \end{bmatrix} \tag{A.9}$$

ii - Coupling condition that corresponds to the continuity at the FSI interface:

$$\partial_{y_s} r_{\lambda} = -\mathbf{D}_s \tag{A.10}$$

and

$$\partial_{y_f} r_{\lambda} = \begin{bmatrix} \mathbf{0} & \mathbf{D}_f & \mathbf{0} \end{bmatrix} \tag{A.11}$$

iii - Structural problem solved with a FEM method:

$$\partial_{y_s} r_s = \mathbf{M}_s \tag{A.12}$$

and

$$\partial_{\lambda} r_s = -\mathbf{D}_s^T \tag{A.13}$$

Combining the given corresponding values of Jacobian computed for all these equations leads to the corresponding value of the contraction constant α, that can be written after simplification:

$$\alpha = \max_n \left\| \left[\partial_{y_s} r_s\right]^{-1} \partial_{\lambda} r_s \left[\partial_{y_f} r_{\lambda} \left[\partial_{y_f} r_f\right]^{-1} \partial_{\lambda} r_f \right]^{-1} \partial_{y_s} r_{\lambda} \right\| =$$
$$\max_n \left\| \mathbf{M}_s^{-1} \mathbf{D}_s^T \left[\mathbf{D}_f \mathbf{M}_f^{-1} \left(\mathbf{1} + \mathbf{B}_f \left(\mathbf{B}_f^T \mathbf{M}_f^{-1} \mathbf{B}_f \right)^{-1} \mathbf{B}_f^T \mathbf{M}_f^{-1} \right) \mathbf{D}_f^T \right]^{-1} \mathbf{D}_s \right\|$$

This criterion shows that the numerical computation is highly linked to the material properties and more generally to the model chosen for the fluid and the structure sub-problems. In the first approximation an estimate for α can be provided by the mass ratio between the fluid $\mathbf{M}_f$ and the solid part $\mathbf{M}_s$ weighted by some geometrical conditions for the field transfer defined in $\mathbf{D}_f$ and $\mathbf{D}_s$. When the mass ratio increase, the scheme can become unstable. The terms in $\mathbf{B}_f$ show the influence of the incompressibility condition, and therefore accounts for the added-mass effect. The fact this behave like and added mass to the fluid problem is clearly stated in the formula. For implicit strategy, to stabilize the iterative DFMT-BGS procedure, Aitken's relaxation can be used as presented in the previous section. The stability of the stabilized algorithm can be then proofed following the steps of [Arnold and Gunther, 2001] for pre-conditioned algorithm. It will not be given herein, and only explored in the numerical examples.

A.3 Proofs for stable error propagation

The proof of the stability error propagation theorem is as follows:

i - The error bound for one iteration of the operator split procedure for an approximation in the neighborhood of the solution is given in **Lemma 1**.

ii - In **Lemma 2**, a recursive application of **Lemma 1** gives a bound for (k) iteration of the operator split procedure.

iii - The proof is concluded by the application of the two Lemmas with suitable arguments.

Lemma 1

Consider $\mathcal{U}_{\gamma_0}$ a neighborhood of the solution $\left(\boldsymbol{x}_f{}^\star, \boldsymbol{x}_s{}^\star, \boldsymbol{y}_s{}^\star\right)$:

$$\mathcal{U}_{\gamma_0} = \left\{ \left(\boldsymbol{x}_f, \boldsymbol{x}_s, \boldsymbol{y}_s\right) \;\middle|\; \left\|\boldsymbol{x}_f - \boldsymbol{x}_f{}^\star\right\| + \|\boldsymbol{x}_s - \boldsymbol{x}_s{}^\star\| + \|\boldsymbol{y}_s - \boldsymbol{y}_s{}^\star\| \leqslant \gamma_0 \right\}$$

There exists $(C, H_0, \gamma_0) \in \mathbb{R}^3_+$ such that:

$$\forall \left((\boldsymbol{x}_f{}^{(0)}, \boldsymbol{x}_s{}^{(0)}, \boldsymbol{y}_s{}^{(0)}), (\tilde{\boldsymbol{x}}_f{}^{(0)}, \tilde{\boldsymbol{y}}_s{}^{(0)}, \tilde{\boldsymbol{\lambda}}^{(0)}) \right) \in \mathcal{U}_{\gamma_0}{}^2, \quad \forall H < H_0,$$

$$\begin{bmatrix} \|\delta\boldsymbol{x}_f{}^{(1)}\| \\ \|\delta\boldsymbol{x}_s{}^{(1)}\| \\ \|\delta\boldsymbol{y}_s{}^{(1)}\| \end{bmatrix} \leqslant \begin{bmatrix} CH & CH & CH \\ CH & CH & CH \\ C & C & \hat{\alpha} + CH \end{bmatrix} \begin{bmatrix} \|\delta\boldsymbol{x}_f{}^{(0)}\| \\ \|\delta\boldsymbol{x}_s{}^{(0)}\| \\ \|\delta\boldsymbol{y}_s{}^{(0)}\| \end{bmatrix} + \begin{bmatrix} \|\delta\boldsymbol{x}_f{}^{(0)}(T_n)\| \\ \|\delta\boldsymbol{x}_s{}^{(0)}(T_n)\| \\ 0 \end{bmatrix}$$

with δ denoting the distance between two approximations, i.e. $\delta x^{(k)} = x^{(k)} - \tilde{x}^{(k)}$ and

$$\hat{\alpha} = \alpha + \mathcal{O}(1)\left(\|\Delta\boldsymbol{x}_f{}^{(0)}\| + \|\Delta\boldsymbol{x}_s{}^{(0)}\| + \|\Delta\boldsymbol{x}_s{}^{(0)}\| + \|\Delta\tilde{\boldsymbol{x}}_f{}^{(0)}\| + \|\Delta\tilde{\boldsymbol{x}}_s{}^{(0)}\| + \|\Delta\tilde{\boldsymbol{x}}_s{}^{(0)}\|\right)$$

Proof of Lemma 1 : By inserting $(\boldsymbol{x}_f{}^{(0)}, \boldsymbol{x}_s{}^{(0)}, \boldsymbol{y}_s{}^{(0)}) \in \mathcal{U}_{\gamma_0}$ in the proposed algorithm, is obtained the evolution equation of the fluid sub-problem (A.6) as the first step of the staggered scheme:

$$\left\{ \begin{array}{rcl} \partial_t \boldsymbol{x}_f{}^{(1)} & = & f_{r_f}(\boldsymbol{x}_f{}^{(1)}, \boldsymbol{x}_s{}^{(0)}, \boldsymbol{y}_f{}^{(1)}) \\ 0 & = & r_f(\boldsymbol{x}_f{}^{(1)}, \boldsymbol{x}_s{}^{(0)}, \boldsymbol{y}_f{}^{(1)}, \boldsymbol{\lambda}^{(1)}) \\ 0 & = & r_\lambda(\boldsymbol{x}_f{}^{(1)}, \boldsymbol{x}_s{}^{(0)}, \boldsymbol{y}_f{}^{(1)}, \boldsymbol{y}_s{}^{(0)}) \\ & & \text{with } \boldsymbol{x}_f{}^{(1)}(T_N) = \boldsymbol{x}_f{}^{(0)}(T_N) \end{array} \right. \tag{A.14}$$

For the second sub-system in (A.7), the evolution of the structure problem under the loading $\boldsymbol{\lambda}^{(k)}$ is considered:

$$\left\{ \begin{array}{rcl} \partial_t \boldsymbol{x}_s{}^{(1)} & = & f_{r_s}(\boldsymbol{x}_f{}^{(1)}, \boldsymbol{x}_s{}^{(1)}, \boldsymbol{y}_s{}^{(1)}) \\ 0 & = & r_s(\boldsymbol{x}_f{}^{(1)}, \boldsymbol{x}_s{}^{(1)}, \boldsymbol{y}_s{}^{(1)}, \boldsymbol{\lambda}^{(1)}) \\ & & \text{with } \boldsymbol{x}_s{}^{(1)}(T_N) = \boldsymbol{x}_s{}^{(0)}(T_N) \end{array} \right. \tag{A.15}$$

The same evolution holds for any other initial value in the neighborhood of the solution: $(\tilde{\boldsymbol{x}}_f{}^{(0)}, \tilde{\boldsymbol{x}}_s{}^{(0)}, \tilde{\boldsymbol{y}}_s{}^{(0)}) \in \mathcal{U}_{\gamma_0}$.

We make a regularity assumption which guarantees that f_{r_f} and f_{r_s} satisfy Lipschitz conditions w.r.t. to their arguments $(\boldsymbol{x}_f, \boldsymbol{x}_s, \boldsymbol{y}_f, \boldsymbol{y}_s)$[2]. So the following bound after one time integration over the window $N+1$ is considered for $\delta \boldsymbol{x}_f{}^{(1)} = \boldsymbol{x}_f{}^{(1)} - \tilde{\boldsymbol{x}}_f{}^{(1)}$ and $\delta \boldsymbol{x}_s{}^{(1)} = \boldsymbol{x}_s{}^{(1)} - \tilde{\boldsymbol{x}}_s{}^{(1)}$:

$$\begin{cases} \|\delta \boldsymbol{x}_f{}^{(1)}\| & \leqslant & \|\delta \boldsymbol{x}_f{}^{(0)}(T_N)\| + \mathcal{O}(H)(\|\delta \boldsymbol{x}_f{}^{(0)}\| + \|\delta \boldsymbol{y}_f{}^{(1)}\|) \\ \|\delta \boldsymbol{x}_s{}^{(1)}\| & \leqslant & \|\delta \boldsymbol{x}_s{}^{(0)}(T_N)\| + \mathcal{O}(H)(\|\delta \boldsymbol{x}_s{}^{(0)}\| + \|\delta \boldsymbol{y}_s{}^{(1)}\|) \end{cases} \tag{A.16}$$

Our goal is now to provide a bound to the difference in the primal unknown that will be imposed at the next iteration $\|\delta \boldsymbol{y}_s{}^{(1)}\|$. For a fixed time t, the algebraic equations are summarized to $F(0) = F(1) = 0$ with:

$$F(\theta) = \begin{bmatrix} r_f(\boldsymbol{x}_f{}^{(1),\theta}, \boldsymbol{x}_s{}^{(0),\theta}, \boldsymbol{y}_f{}^{(1),\theta}, \boldsymbol{\lambda}^{(1),\theta}) \\ r_\lambda(\boldsymbol{x}_f{}^{(1),\theta}, \boldsymbol{x}_s{}^{(0),\theta}, \boldsymbol{y}_f{}^{(1),\theta}, \boldsymbol{y}_s{}^{(0),\theta}) \\ r_s(\boldsymbol{x}_f{}^{(1),\theta}, \boldsymbol{x}_s{}^{(1),\theta}, \boldsymbol{y}_s{}^{(1),\theta}, \boldsymbol{\lambda}^{(1),\theta}) \end{bmatrix}, \quad \theta \in \{0,1\} \tag{A.17}$$

where x^θ can be written as $x^\theta = (1-\theta)x + \theta\tilde{x}$.

The identity $F(1) - F(0) = \int_0^1 F'(\theta)\,\mathrm{d}\theta = 0$ further gives:

$$\int_0^1 \overbrace{\left(\begin{bmatrix} \partial_{\boldsymbol{y}_f} r_f & \partial_\lambda r_f & 0 \\ \partial_{\boldsymbol{y}_f} r_\lambda & 0 & 0 \\ 0 & \partial_\lambda r_s & \partial_{\boldsymbol{y}_s} r_s \end{bmatrix} \begin{bmatrix} \delta \boldsymbol{y}_f{}^{(1)} \\ \delta \boldsymbol{\lambda}^{(1)} \\ \delta \boldsymbol{y}_s{}^{(1)} \end{bmatrix} + \begin{bmatrix} 0 \\ \partial_{\boldsymbol{y}_s} r_\lambda \\ 0 \end{bmatrix} \delta \boldsymbol{y}_s{}^{(0)} \right)}^{\text{independent of } \theta} \mathrm{d}\theta$$

$$+ \mathcal{O}(1) \left(\|\delta \boldsymbol{x}_s{}^{(0)}\| + \|\delta \boldsymbol{x}_f{}^{(1)}\| + \|\delta \boldsymbol{x}_s{}^{(1)}\| \right) = 0$$

In the equation above the arguments $\boldsymbol{x}_f{}^{(1)}$ and $\boldsymbol{x}_s{}^{(1)}$ of the Jacobian that are placed in the neighborhood $\mathcal{U}_{\gamma_0}$ of size $\mathcal{O}(\gamma_0)$ of the solution $(\boldsymbol{x}_f{}^\star, \boldsymbol{x}_s{}^\star)$ are neglected. If γ_0 is sufficiently small, then we can easily solve the previous equation, provided the algebraic equations give a non-singular matrix[3].

$$\begin{aligned} - & \begin{bmatrix} \partial_{\boldsymbol{y}_f} r_f & \partial_\lambda r_f & 0 \\ \partial_{\boldsymbol{y}_f} r_\lambda & 0 & 0 \\ 0 & \partial_\lambda r_s & \partial_{\boldsymbol{y}_s} r_s \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ \partial_{\boldsymbol{y}_s} r_\lambda \\ 0 \end{bmatrix} \\ = & \begin{bmatrix} & \\ & \\ \left[\partial_{\boldsymbol{y}_s} r_s\right]^{-1} \partial_\lambda r_s \left[\partial_{\boldsymbol{y}_f} r_\lambda \left[\partial_{\boldsymbol{y}_f} r_f\right]^{-1} \partial_\lambda r_f\right]^{-1} \partial_{\boldsymbol{y}_s} r_\lambda \end{bmatrix} \end{aligned} \tag{A.18}$$

thus the contractivity coefficient α can be introduced, defined as:

$$\alpha = \max_n \left\| \left[\partial_{\boldsymbol{y}_s} r_s\right]^{-1} \partial_\lambda r_s \left[\partial_{\boldsymbol{y}_f} r_\lambda \left[\partial_{\boldsymbol{y}_f} r_f\right]^{-1} \partial_\lambda r_f\right]^{-1} \partial_{\boldsymbol{y}_s} r_\lambda \right\|$$

[2] On the regularity assumption, see the remark at the end of this proof.

[3] This requires non-singular Jacobians, see [Arnold and Gunther, 2001, Brenan et al., 1996] for more details

Using this inverse calculation in A.18 gives the following bounds for the difference between the two coupled problems after one iteration:

$$\begin{aligned}\|\delta \boldsymbol{y}_f{}^{(1)}\| &\leqslant \mathcal{O}(1)\Big(\|\delta \boldsymbol{x}_f{}^{(0)}\|+\|\delta \boldsymbol{x}_s{}^{(0)}\|+\|\delta \boldsymbol{y}_s{}^{(0)}\|+\|\delta \boldsymbol{x}_f{}^{(1)}\|+\|\delta \boldsymbol{x}_s{}^{(1)}\|\Big)\\ \|\delta \boldsymbol{y}_s{}^{(1)}\| &\leqslant \hat{\alpha}\|\delta \boldsymbol{y}_s{}^{(0)}\|+\mathcal{O}(1)\Big(\|\delta \boldsymbol{x}_f{}^{(0)}\|+\|\delta \boldsymbol{x}_s{}^{(0)}\|+\|\delta \boldsymbol{x}_f{}^{(1)}\|+\|\delta \boldsymbol{x}_s{}^{(1)}\|\Big)\end{aligned}$$

where:

$$\hat{\alpha}=\alpha+\mathcal{O}(1)\Big(\|\Delta \boldsymbol{x}_f{}^{(0)}\|+\|\Delta \boldsymbol{x}_s{}^{(0)}\|+\|\Delta \boldsymbol{\lambda}^{(0)}\|+\|\Delta \tilde{\boldsymbol{x}}_f{}^{(0)}\|+\|\Delta \tilde{\boldsymbol{x}}_s{}^{(0)}\|+\|\Delta \tilde{\boldsymbol{\lambda}}^{(0)}\|\Big)$$

Inserting the inequality above in the Lipschitz condition Eq. (A.16) gives:

$$\begin{aligned}\|\delta \boldsymbol{x}_f{}^{(1)}\| &\leqslant \mathcal{O}(H)\left(\|\delta \boldsymbol{x}_f{}^{(0)}\|+\|\delta \boldsymbol{x}_s{}^{(0)}\|+\|\delta \boldsymbol{y}_s{}^{(0)}\|\right)+\|\delta \boldsymbol{x}_f{}^{(0)}(T_N)\|\\ \|\delta \boldsymbol{x}_s{}^{(1)}\| &\leqslant \mathcal{O}(H)\left(\|\delta \boldsymbol{x}_f{}^{(0)}\|+\|\delta \boldsymbol{x}_s{}^{(0)}\|+\|\delta \boldsymbol{y}_s{}^{(0)}\|\right)+\|\delta \boldsymbol{x}_s{}^{(0)}(T_N)\|\\ \|\delta \boldsymbol{y}_s{}^{(1)}\| &\leqslant (\hat{\alpha}+\mathcal{O}(H))\,\|\delta \boldsymbol{y}_f{}^{(0)}\|+\mathcal{O}(1)\left(\|\delta \boldsymbol{x}_f{}^{(0)}\|+\|\delta \boldsymbol{x}_s{}^{(0)}\|\right)\end{aligned}$$

Rewriting the equation above in matrix form completes the proof of **Lemma 1**. ■

Lemma 2

Provided that the assumptions of the previous Lemma are satisfied and assume that $\hat{\alpha}<1$ and $C>\hat{\alpha}$, one can write:

$$\exists \hat{C}\in\mathbb{R}_+ \qquad \text{such that} \qquad \forall k>1, \forall H\leqslant H_0$$

$$\begin{bmatrix}\|\delta \boldsymbol{x}_f{}^{(k)}\|\\ \|\delta \boldsymbol{x}_s{}^{(k)}\|\\ \|\delta \boldsymbol{y}_s{}^{(k)}\|\end{bmatrix} \leqslant \begin{bmatrix}1+\hat{C}H\|\delta \boldsymbol{x}_f{}^{(0)}(T_N)\|\\ 1+\hat{C}H\|\delta \boldsymbol{x}_s{}^{(0)}(T_N)\|\\ \hat{C}\end{bmatrix}+ \begin{bmatrix}\hat{C}(4\hat{C}+1)H\hat{\mu}^{k-2} & \hat{C}(4\hat{C}+1)H\hat{\mu}^{k-2} & 4\hat{C}H\hat{\mu}^{k-1}\\ \hat{C}(4\hat{C}+1)H\hat{\mu}^{k-2} & \hat{C}(4\hat{C}+1)H\hat{\mu}^{k-2} & 4\hat{C}H\hat{\mu}^{k-1}\\ 4\hat{C}\hat{\mu}^{k-1} & 4\hat{C}\hat{\mu}^{k-1} & \hat{\mu}^k+(\hat{\mu}-\hat{\alpha})^k\end{bmatrix}\begin{bmatrix}\|\delta \boldsymbol{x}_f{}^{(0)}\|\\ \|\delta \boldsymbol{x}_s{}^{(0)}\|\\ \|\delta \boldsymbol{y}_s{}^{(0)}\|\end{bmatrix}$$

with $\hat{\mu}=\hat{\alpha}+\frac{2CH}{\frac{\hat{\alpha}}{2C}+\sqrt{H}}$

Proof of Lemma 2 : Although the successive values of $(\boldsymbol{x}_f{}^{(k)}, \boldsymbol{x}_s{}^{(k)}, \boldsymbol{y}_s{}^{(k)})$ remain in the neighborhood $\mathcal{U}_{\gamma_0}$ of the solution, **Lemma 1** shows that iteration error is mainly governed by the matrix:

$$\mathbf{J}=\begin{bmatrix}CH & CH & CH\\ CH & CH & CH\\ C & C & \hat{\alpha}+CH\end{bmatrix} \tag{A.19}$$

Recursive application of **Lemma 1** leads to:

$$\begin{bmatrix}\|\delta \boldsymbol{x}_f{}^{(k)}\|\\ \|\delta \boldsymbol{x}_s{}^{(k)}\|\\ \|\delta \boldsymbol{y}_s{}^{(k)}\|\end{bmatrix} \leqslant \mathbf{J}^k\begin{bmatrix}\|\delta \boldsymbol{x}_f{}^{(0)}\|\\ \|\delta \boldsymbol{x}_s{}^{(0)}\|\\ \|\delta \boldsymbol{y}_s{}^{(0)}\|\end{bmatrix}+\sum_{i=0}^{k-1}\mathbf{J}^i\begin{bmatrix}\|\delta\Delta \boldsymbol{x}_f{}^{(0)}(T_N)\|\\ \|\delta\Delta \boldsymbol{x}_s{}^{(0)}(T_N)\|\\ 0\end{bmatrix} \tag{A.20}$$

Hence, the goal is to express in a relatively simple way the bounds for the elements of $\mathbf{J}^k$ and for the first column of $\sum \mathbf{J}^i$. The classical method used to obtain such bounds seeks to first transform the matrix into corresponding diagonal form $\mathbf{J}$ and then to bound the power of the eigenvalues. This leads to tedious but straight-forward calculations; for more details the reader is invited to consider [Arnold and Gunther, 2001], Lemma 3.2. ■

Proof of Theorem 1 : The errors on the interest variables like $\epsilon_{\boldsymbol{x}_f,N}$ on $\boldsymbol{x}_f$ for the fluid part, $\epsilon_{\boldsymbol{x}_s,N}$ on $\boldsymbol{x}_s$ for the structure part, and $\epsilon_{\boldsymbol{y}_s,N}$ on $\boldsymbol{y}_s$ used for the coupling, are split in two terms: $e_{.,N+1}$ representing error propagation from one window N ($t \in [T_{N-1}, T_N]$) to the next $N+1$ ($t \in [T_N, T_{N+1}]$); $d_{.,N+1}$ corresponds to local error contribution on the window of interest $N+1$.

The proof is organized as follows: applying **Lemma 2** with suitable arguments, yields estimates for error propagation e (see proof (i)) and local error d (see proof (ii)). Then these estimates are combined to bounds set for the global errors ϵ (see proof(iii)). We can then show by induction that the global error bound is always verified (see proof (iv)). The constants μ and α are explicitly stated at the end of the proof of **Theorem 1** (v).

i - Estimate of propagation error contribution is the first part of the proof, where **Lemma 2** with the following suitable arguments is applied:

$$\begin{bmatrix} \tilde{\boldsymbol{x}_f}^{(0)} \\ \tilde{\boldsymbol{x}_s}^{(0)} \\ \tilde{\boldsymbol{y}_s}^{(0)} \end{bmatrix} \longleftarrow \begin{bmatrix} \boldsymbol{x}_{f\,N}^{(k_{\max})} \\ \boldsymbol{x}_{s\,N}^{(k_{\max})} \\ \boldsymbol{y}_{s\,N}^{(k_{\max})} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{x}_f^{(0)} \\ \boldsymbol{x}_s^{(0)} \\ \boldsymbol{y}_s^{(0)} \end{bmatrix} \longleftarrow \begin{bmatrix} \boldsymbol{x}_{f\,N}^{\star} \\ \boldsymbol{x}_{s\,N}^{\star} \\ \boldsymbol{y}_{s\,N}^{\star} \end{bmatrix} \tag{A.21}$$

The symbol, '$\longleftarrow$' indicates the initial guesses (obtained with zeroth, first or second order predictors) from one window N to the next one $N+1$. For instance, in the simplest predictor is the constant function that leads to an error of size $\mathcal{O}(H)$:

$$(\boldsymbol{x}_{f\,N+1}^{(0)}, \boldsymbol{x}_{s\,N+1}^{(0)}, \boldsymbol{y}_{s\,N+1}^{(0)}) = (\boldsymbol{x}_{f\,N}^{(k_{\max})}, \boldsymbol{x}_{s\,N}^{(k_{\max})}, 0)$$

The values $(\boldsymbol{x}_{f\,N}^{(k_{\max})}, \boldsymbol{x}_{s\,N}^{(k_{\max})}, \boldsymbol{y}_{s\,N}^{(k_{\max})})$, in equation (A.21) above, obtained by numerical integration on the previous window N, can be considered as the best representation of the exact solution of the problem $(\boldsymbol{x}_{f\,N}^{\star}, \boldsymbol{x}_{s\,N}^{\star}, \boldsymbol{y}_{s\,N}^{\star})$ on this window. The choice of initial values (A.21) gives by definition the propagation error in the $N+1$-th windows for the (k)-th iteration: $\delta\boldsymbol{x}_f^{(k)} = e_{\boldsymbol{x}_f,N+1}$, $\delta\boldsymbol{x}_s^{(k)} = e_{\boldsymbol{x}_s,N+1}$ and $\delta\boldsymbol{y}_s^{(k)} = e_{\boldsymbol{y}_s,N+1}$.

It is assumed that the chosen initial guess operator $\longleftarrow$ satisfies

Lipschitz condition:

$$\exists L^* \in \mathbb{R} \quad \text{such that} \quad \begin{cases} \|\Delta \boldsymbol{x}_f{}^{(0)}\| & \leqslant & L^* \|\epsilon_{\boldsymbol{x}_f,N}\| \\ \|\Delta \boldsymbol{x}_s{}^{(0)}\| & \leqslant & L^* \|\epsilon_{\boldsymbol{x}_s,N}\| \\ \|\Delta \boldsymbol{y}_s{}^{(0)}\| & \leqslant & L^* \|\epsilon_{\boldsymbol{y}_s,N}\| \end{cases} \tag{A.22}$$

It will extrapolate $\boldsymbol{x}_f$ and $\boldsymbol{x}_s$ continuously from one window to another: $\|\delta \boldsymbol{x}_f{}^{(0)}(T_N)\| \leqslant \|\epsilon_{\boldsymbol{x}_f,N}\|$ and $\|\delta \boldsymbol{x}_s{}^{(0)}(T_N)\| \leqslant \|\epsilon_{\boldsymbol{x}_s,N}\|$. Therefore, the application of **Lemma 2** to the (k)-th iteration on window N with $\alpha < 1$ and a window size H small enough so that $\mu < 1$ leads to:

$$\begin{bmatrix} \|e_{\boldsymbol{x}_f,N+1}\| \\ \|e_{\boldsymbol{x}_s,N+1}\| \\ \|e_{\boldsymbol{y}_s,N+1}\| \end{bmatrix} \leqslant \begin{bmatrix} 1+C_1^*H & 1+C_1^*H & C_1^*H \\ 1+C_1^*H & 1+C_1^*H & C_1^*H \\ C_1^* & C_1^* & \alpha^* \end{bmatrix} \begin{bmatrix} \|\epsilon_{\boldsymbol{x}_f,N}\| \\ \|\epsilon_{\boldsymbol{x}_s,N}\| \\ \|\epsilon_{\boldsymbol{y}_s,N}\| \end{bmatrix} \tag{A.23}$$

with $C_1^* \in \mathbb{R}_+$ and $\alpha^* := L^*(\hat{\mu}^k + (\hat{\mu} - \hat{\alpha})^k)$.

ii - Estimate of local error contribution consists again in applying **Lemma 2** with suitable arguments:

$$\begin{bmatrix} \tilde{\boldsymbol{x}}_f{}^{(0)} \\ \tilde{\boldsymbol{x}}_s{}^{(0)} \\ \tilde{\boldsymbol{y}}_s{}^{(0)} \end{bmatrix} \longleftarrow \begin{bmatrix} \boldsymbol{x}_f{}^{\star}_N \\ \boldsymbol{x}_s{}^{\star}_N \\ \boldsymbol{y}_s{}^{\star}_N \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} \boldsymbol{x}_f{}^{(0)} \\ \boldsymbol{x}_s{}^{(0)} \\ \boldsymbol{y}_s{}^{(0)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_f{}^{\star}_{N+1} \\ \boldsymbol{x}_s{}^{\star}_{N+1} \\ \boldsymbol{y}_s{}^{\star}_{N+1} \end{bmatrix} \tag{A.24}$$

Since the solution of the problem $(\boldsymbol{x}_f{}^{\star}_n, \boldsymbol{x}_s{}^{\star}_n, \boldsymbol{\lambda}^{\star}_n)$ is a fixed-point of the iteration sequence, the local error contributions are measured by $(\delta \boldsymbol{x}_f{}^{(k)}, \delta \boldsymbol{x}_s{}^{(k)}, \delta \boldsymbol{y}_s{}^{(k)}) = (d_{\boldsymbol{x}_f,N+1}, d_{\boldsymbol{x}_s,N+1}, d_{\boldsymbol{\lambda},N+1})$. Furthermore, the use of the solution of the problem as the initialization of the iterative sequence yields $(\delta \boldsymbol{x}_f{}^{(0)}, \delta \boldsymbol{x}_s{}^{(0)}, \delta \boldsymbol{y}_s{}^{(0)}) = (\Delta \boldsymbol{x}_f{}^{(0)}, \Delta \boldsymbol{x}_s{}^{(0)}, \Delta \boldsymbol{y}_s{}^{(0)})$, $\tilde{\boldsymbol{x}}_f{}^{(0)}(T_N) - \boldsymbol{x}_f{}^{(0)}(T_N) = 0$ and $\tilde{\boldsymbol{x}}_s{}^{(0)}(T_N) - \boldsymbol{x}_s{}^{(0)}(T_N) = 0$. Thus the application of **Lemma 2** then gives:

$$\begin{bmatrix} \|d_{\boldsymbol{x}_f,N+1}\| \\ \|d_{\boldsymbol{x}_s,N+1}\| \\ \|d_{\boldsymbol{y}_s,N+1}\| \end{bmatrix} \leqslant \\ \left(\hat{\mu}^{k-2}\left(\|\Delta \boldsymbol{x}_f{}^{(0)}\| + \|\Delta \boldsymbol{x}_s{}^{(0)}\|\right) + \hat{\mu}^{k-1}\|\Delta \boldsymbol{\lambda}^{(0)}\|\right) \begin{bmatrix} C_2^*H \\ C_2^*H \\ C_2^* \end{bmatrix} \tag{A.25}$$

with a positive constant C_2^*.

iii - these estimates are combined with the bounds set for the global errors $\|\boldsymbol{\epsilon}_{N+1}\| \leqslant \|\mathbf{e}_{N+1}\| + \|\mathbf{d}_{N+1}\|$:

$$\begin{bmatrix} \|\epsilon_{\boldsymbol{x}_f,N+1}\| \\ \|\epsilon_{\boldsymbol{x}_s,N+1}\| \\ \|\epsilon_{\boldsymbol{y}_s,N+1}\| \end{bmatrix} \leqslant \begin{bmatrix} 1+C_1^*H & 1+C_1^*H & C_1^*H \\ 1+C_1^*H & 1+C_1H^* & C_1^*H \\ C_1^* & C_1^* & \alpha^* \end{bmatrix} \begin{bmatrix} \|\epsilon_{\boldsymbol{x}_f,N}\| \\ \|\epsilon_{\boldsymbol{x}_s,N}\| \\ \|\epsilon_{\boldsymbol{y}_s,N}\| \end{bmatrix} \\ + \left(\hat{\mu}^{k-2}\left(\|\Delta \boldsymbol{x}_f{}^{(0)}\| + \|\Delta \boldsymbol{x}_s{}^{(0)}\|\right) + \hat{\mu}^{k-1}\|\Delta \boldsymbol{y}_s{}^{(0)}\|\right) \begin{bmatrix} C_2^*H \\ C_2^*H \\ C_2^* \end{bmatrix} \tag{A.26}$$

If the contraction condition $\alpha^* < 1$ is fulfilled, the behavior of such coupled error recursions is known (see [Deuflhard et al., 1987], Lemma 2) and can be written as follows:

$$\begin{aligned} &\max_N \left(\|\epsilon_{\boldsymbol{x}_f,N}\| + \|\epsilon_{\boldsymbol{x}_s,N}\| + \|\epsilon_{\boldsymbol{y}_s,N}\| \right) \\ &\leqslant C \cdot \max_N \left(\hat{\mu}^{k-2} \left(\|\Delta \boldsymbol{x}_f{}^{(0)}\| + \|\Delta \boldsymbol{x}_s{}^{(0)}\| \right) + \hat{\mu}^{k-1} \|\Delta \boldsymbol{y}_s{}^{(0)}\| \right) \end{aligned} \tag{A.27}$$

with a positive constant C.

As the total error is the expressed by the sum $\|\epsilon_{\boldsymbol{x}_f,N}\| + \|\epsilon_{\boldsymbol{x}_s,N}\| + \|\epsilon_{\boldsymbol{y}_s,N}\|$, the last expression completes the proof of **Theorem 1**.

iv - As the initial guess operator $\longleftarrow$ leads to initial errors $\|\Delta \boldsymbol{x}_f{}^{(0)}\|$, $\|\Delta \boldsymbol{x}_s{}^{(0)}\|$ and $\|\Delta \boldsymbol{y}_s{}^{(0)}\|$ of size $\mathcal{O}(H)$, the right-hand side of the last equation remains bounded for all $H < H_0$, if H_0 is sufficiently small:

$$\begin{aligned} &\max_N \left(\|\epsilon_{\boldsymbol{x}_f,N}\| + \|\epsilon_{\boldsymbol{x}_s,N}\| + \|\epsilon_{\boldsymbol{y}_s,N}\| \right) \\ &\leqslant C \cdot \max_N \left(\hat{\mu}^{k-2} \left(\|\Delta \boldsymbol{x}_f{}^{(0)}\| + \|\Delta \boldsymbol{x}_s{}^{(0)}\| \right) + \hat{\mu}^{k-1} \|\Delta \boldsymbol{\lambda}^{(0)}\| \right) \\ &\leqslant \gamma_0 \end{aligned} \tag{A.28}$$

This further shows that errors is bounded by γ_0, and that approximate numerical solution remains in the neighborhood $\mathcal{U}_{\gamma_0}$ of the solution.

v - The constants $\hat{\alpha}$ and $\hat{\mu}$ which appear in proofs of **Lemma 1** and **Lemma 2** are given as follows:

$$\hat{\alpha} = \alpha + \mathcal{O}(1) \max_n (\|\epsilon_{\boldsymbol{x}_f,N}\| + \|\epsilon_{\boldsymbol{x}_s,N}\| + \|\epsilon_{\boldsymbol{\lambda},N}\|) + \mathcal{O}(H) = \alpha + \mathcal{O}(H)$$

and

$$\hat{\mu} = \hat{\alpha} + \mathcal{O}(H)$$

These results give the following order of magnitude for α^*:

$$\alpha^* = L^*(\hat{\mu}^k + (\hat{\mu} - \hat{\alpha})^k) = L^*((\alpha + \mathcal{O}(H)^k + \mathcal{O}(H^k)) \tag{A.29}$$

The expression above confirms the criterion of stable error propagation for $\alpha^* < 1$. For the iterative DFMT-BGS with an appropriate guess operator, the stability is guaranteed when $\alpha < 1$ and when the window size is small enough. ■

Bibliography

[Adams et al., 2004] Adams, M. F., Bayraktar, H. H., Keaveny, T. M., and Panayiotis, P. (2004). Ultrascalable implicit finite element analyses in solid mechanics with over a half billion degrees of freedom. SC2004 High performance computing, networking and storage conference, Pittsburg, PA. (Cited on page 63.)

[Arnold, 2001] Arnold, M. (2001). Constraint partitioning in dynamic iteration methods. *Zeitschrift fur Angewandte Mathematik und Mechanik*, 81:735–738. (Cited on page 131.)

[Arnold and Gunther, 2001] Arnold, M. and Gunther, M. (2001). Preconditioned dynamic iteration for coupled differential-algebraic systems. *BIT Numer. Math.*, 41:1–25. (Cited on pages 48, 50, 129, 131, 132, 134 and 136.)

[Austruy, 2008] Austruy, C. (2008). Approches multi-échelles et multi-physiques pour quantifier l'amortissement d'une vague par des obstacles. Mémoire de Master 2-R MIS, parcours Génie-civil, École Normale Supérieure de Cachan, Cachan, France. Encadrants : J.-B. Colliat, C. Kassiotis. Mention TB, Rang 1. (Cited on pages 65 and 91.)

[Austruy et al., 2008] Austruy, C., Kassiotis, C., Colliat, J.-B., Ibrahimbegović, A., Matthies, H. G., and Dias, F. (2008). A multiscale and multiphysic approach to quantify waves damping by structures. In Ibrahimbegović, A. and Zlatar, M., editors, *NATO–ARW 983112 Damage assessments and reconstruction after natural disasters and previous military activities*, Sarajevo, Bosnia-Herzegovina. (Cited on pages 65 and 88.)

[Barcelos et al., 2006] Barcelos, M., Bavestrello, H., and Maute, K. (2006). A Schur–Newton–Krylov solver for steady-state aeroelastic analysis and design sensitivity analysis. *Computer Methods in Applied Mechanics and Engineering*, 195:2050–2069. (Cited on page 58.)

[Barton, 1998] Barton, I. E. (1998). Comparison of SIMPLE- and PISO-type algorithms for transient flows. *International Journal for Numerical Methods in Fluid*, 26(4):459–483. (Cited on page 15.)

[Bathe and Zhang, 2009] Bathe, K.-J. and Zhang, H. (2009). A mesh adaptivity procedure for CFD and fluid-structure interactions. *Computers and Structures*, 87(11-12):604–617. (Cited on page 98.)

[Baudille and Biancolini, 2006] Baudille, R. and Biancolini, M. E. (2006). Modelling FSI problems in FLUENT: a general purpose approach by means of UDF programming. *Unknwown*. http://www.colorado.edu/engineering/CAS/courses.d/FSI.d/FSI.CISM.d/Ross.Thesis.pdf.

[Becker and Rannacher, 2003] Becker, R. and Rannacher, R. (2003). An optimal control approach to a posteriori error estimation in finite element methods. *Acta numerica*, 10:1–102. (Cited on page 7.)

[Beckert and Wendland, 2001] Beckert, A. and Wendland, H. (2001). Multivariate interpolation for fluid-structure-interaction problems using radial basis functions. *Aerospace Science and Technology*, 5(2):125–134. (Cited on pages 89 and 91.)

[Belytschko et al., 1979] Belytschko, T., Yen, H. J., and Mullen, R. (1979). Mixed methods for time integration. *Computer Methods in Applied Mechanics and Engineering*, 17(18):259–275. (Cited on page 37.)

[Bernardi et al., 1990] Bernardi, C., Maday, Y., and Patera, A. T. (1990). A new nonconforming approach to domain decomposition: the mortar element method. Technical report, Université Pierre at Marie Curie, Paris, France. (Cited on page 38.)

[Berti, 2002] Berti, G. (2002). *Generic Software Components for Scientific Computing*. Ph.D. Thesis, Technische Universität Cottbus, Germany. (Cited on page 64.)

[Bouche et al., 2006] Bouche, D., Ghidaglia, J. M., and Pascal, F. (2006). Error estimate and the geometric corrector for the upwind finite volume method applied to the linear advection equation. *SIAM Journal on Numerical Analysis*, 43(2):578–603. (Cited on page 7.)

[Brenan et al., 1996] Brenan, K. E., Campbell, S. L. V., and Petzold, L. R. (1996). *Numerical solution of initial-value problems in Differential-Algebraic Equations*. Society for Industrial and Applied Mathematics. (Cited on page 134.)

[Brooks and Hughes, 1990] Brooks, A. N. and Hughes, T. J. R. (1990). Streamline upwind/Petrov-Galerkin formulations for convection dominated flows with particular emphasis on the incompressible Navier-Stokes equations. *Computer methods in applied mechanics and engineering*, pages 199–259. (Cited on pages 7 and 37.)

[Broy et al., 1998] Broy, M., Deimel, A., Henn, J., Koskimies, K., Plášil, F., Pomberger, G., Pree, W., Stal, M., and Szyperski, C. (1998). What characterizes a (software) component? *Software – Concept & Tools*, 19:49–56. (Cited on page 65.)

[Bruneau and Saad, 2006] Bruneau, C.-H. and Saad, M. (2006). The 2D lid-driven cavity problem revisited. *Computers and Fluids*, 35:326–348. (Cited on page 96.)

[Bügling, 2006] Bügling, B. (2006). *The Component Template Library Protocol and its Java Implementation.* Master Thesis, Technische Universität Braunschweig, Institut for Scientific Computing. http://www.wire.tu-bs.de/forschung/projekte/ctl/files/ctl_spec.pdf. (Cited on page 66.)

[Bullock et al., 2007] Bullock, G., Obhrai, C., Peregrine, D., and Bredmose, H. (2007). Violent breaking wave impacts. Part I: Results from large-scale regular wave tests on vertical and sloping walls. *Coastal Engineering*, 54(8):602–617. (Cited on page 29.)

[Causin et al., 2005] Causin, P., Gerbeau, J.-F., and Nobile, F. (2005). Added-mass effect in the design of partitioned algorithms for fluid-structure problems. *Computer Methods in Applied Mechanics and Engineering*, 194(42-44):4506–4527. (Cited on pages 3, 47 and 103.)

[Chung and Hulbert, 1994] Chung, J. and Hulbert, G. M. (1994). A family of single-step houbolt time integration algorithms for structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 118(1-2):1–11. (Cited on page 10.)

[Clough, 1960] Clough, R. W. (1960). The finite element method in plane stress analysis. ASCE conference on electronic computation, Pittsburg, PA. (Cited on page 1.)

[Comité Européen de Normalisation, 2009] Comité Européen de Normalisation (1971–2009). Eurocode home page. http://eurocodes.jrc.ec.europa.eu/. (Cited on page 1.)

[Coulange, 1998] Coulange, B. (1998). *Software reuse.* Springer Verlag, London. (Cited on page 64.)

[de Boer et al., 2007] de Boer, A., van Zuijlen, A. H., and Bijl, H. (2007). Review of coupling methods for non-matching meshes. *Computer Methods in Applied Mechanics and Engineering*, 196(8):1515–1525. (Cited on page 88.)

[Demirdžić and Perić, 1988] Demirdžić, I. and Perić, M. (1988). Space conservation law in finite volume calculations of fluid flow. *International Journal for Numerical Methods in Fluid*, 8(9). (Cited on pages 8, 25 and 45.)

[Deparis, 2004] Deparis, S. (2004). *Numerical Analysis of Axisymmetric Flows and Methods for Fluid-Structure Interaction Arising in Blood Flow Simulation.* Ph.D. Thesis, École Polytechnique Fédérale de Lausanne, Swiss. (Cited on pages 1 and 40.)

[Deparis et al., 2006] Deparis, S., Discacciati, M., Fourestey, G., and Quarteroni, A. (2006). Fluid-structure algorithms based on Steklov-Poincaré operators. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5797–5812. (Cited on pages 41, 53 and 58.)

[Dettmer and Perić, 2003] Dettmer, W. G. and Perić, D. (2003). An analysis of the time integration algorithms for the Finite Element solution of incompressible Navier-Stokes equations based on a stabilised formulation. *Computer Methods in Applied Mechanics and Engineering*, 192:1177–1226. (Cited on pages 10 and 37.)

[Dettmer and Perić, 2006] Dettmer, W. G. and Perić, D. (2006). A computational framework for fluid–structure interaction: finite element formulation and applications. *Computer Methods in Applied Mechanics and Engineering*, 195(41-43):5754–5779. (Cited on pages 106 and 109.)

[Dettmer and Perić, 2007] Dettmer, W. G. and Perić, D. (2007). A fully implicit computational strategy for strongly coupled fluid-solid interaction. *Archives of Computational Methods in Engineering*, 14:205–247. (Cited on pages 37, 95, 106, 109 and 111.)

[Dettmer and Perić, 2008] Dettmer, W. G. and Perić, D. (2008). On the coupling between fluid flow and mesh motion in the modelling of fluid-structure interaction. *Computational Mechanics*, 43(1):81–90. (Cited on page 58.)

[Deuflhard et al., 1987] Deuflhard, P., Hairer, E., and Zugck, J. (1987). One-step and extrapolation methods for differential-algebraic systems. *Numerische Mathematik*, 51:501–516. (Cited on page 138.)

[Dias and Dutykh, 2006] Dias, F. and Dutykh, D. (2006). Dynamics of tsunami waves. *NATO Advanced Research Workshop, ARW-981641*, pages 35–64. (Cited on page 116.)

[Dias et al., 2009] Dias, F., Dutykh, D., and Ghidaglia, J.-M. (2009). A two-fluid model for violent aerated flows. *Computers and Fluids*, In Press, Accepted Manuscript. (Cited on page 28.)

[Dutykh, 2008] Dutykh, D. (2008). *Modélisation Mathématique des Tsunamis*. Thèse de Doctorat, Centre de Mathématiques et Leurs Applications, École Normale Supérieure de Cachan, France. (Cited on pages 3 and 28.)

[Dutykh and Mitsotakis, 2009] Dutykh, D. and Mitsotakis, D. (2009). On the relevance of the dam break problem in the context of nonlinear shallow water equations. *Discrete and Continous Dynamical System – Series A*, Accepted. (Cited on pages 26 and 29.)

[Farhat et al., 2001] Farhat, C., Geuzaine, P., and Grandmont, C. (2001). The discrete geometric conservation law and the nonlinear stability of ale schemes for the solution of flow problems on moving grids. *Journal of Computational Physics*, 174(2):669–694. (Cited on pages 25 and 45.)

[Farhat and Lesoinne, 2000] Farhat, C. and Lesoinne, M. (2000). Two efficient staggered algorithms for the serial and parallel solution of three-dimensional nonlinear transient aeroelastic problems. *Computer Methods in Applied Mechanics and Engineering*, 182:499–515. (Cited on page 45.)

[Farhat et al., 1995] Farhat, C., Lesoinne, M., and Maman, N. (1995). Mixed explicit/implicit time integration of coupled aeroelastic problems: three-field formulation, geometric conservation and distributed solution. *International Journal for Numerical Methods in Engineering*, 21(10). (Cited on pages 41 and 46.)

[Farhat and Roux, 1994] Farhat, C. and Roux, F.-X. (1994). Implicit parallel processing in structural mechanics. *Computational Mechanics Advances*, 2(1):1–124. (Cited on page 79.)

[Felippa and Park, 2004] Felippa, C. and Park, K. (2004). Synthesis tools for structural dynamics and partitioned analysis of coupled systems. *NATO Advanced Research Workshop (eds. A. Ibrahimbegović and B. Brank)*, pages 50–111. (Cited on pages 37, 50 and 88.)

[Felippa et al., 1977] Felippa, C. A., Park, K. C., and de Runtz, J. A. (1977). Stabilization of staggered solution procedures for fluid-structure interaction analysis. In *Computational methods for fluid-structure interaction problems*, pages 95–124. (Cited on pages 37 and 47.)

[Fernández et al., 2008] Fernández, M. Á., Gerbeau, J.-F., Gloria, A., and Vidrascu, M. (2008). Domain decomposition based Newton methods for fluid-structure interaction problems. In *ESAIM: Proceedings*, volume 22, pages 67–82. edpsciences.org. (Cited on page 41.)

[Fernández et al., 2007] Fernández, M. Á., Gerbeau, J.-F., and Grandmont, C. (2007). A projection semi-implicit scheme for the coupling of an elastic structure with an incompressible fluid. *International Journal for Numerical Methods in Engineering*, 69(4):794–821. (Cited on page 58.)

[Fernández and Moubachir, 2005] Fernández, M. Á. and Moubachir, M. (2005). A Newton method using exact Jacobians for solving fluid–structure coupling. *Computers and Structures*, 83(2-3):127–142. (Cited on pages 40 and 58.)

[Ferziger and Perić, 1996] Ferziger, J. H. and Perić, M. (1996). Further discussion of numerical errors in CFD. *International Journal for Numerical Methods in Fluid*, 23:1263–1274. (Cited on page 52.)

[Ferziger and Perić, 2002] Ferziger, J. H. and Perić, M. (2002). *Computational Methods for Fluid Dynamics*. Springler-Verlag, Berlin, Germany, 3^{rd} edition. (Cited on pages 2, 7, 13, 14, 15, 16, 37, 71, 75, 96 and 98.)

[Feyel and Chaboche, 2001] Feyel, F. and Chaboche, J.-L. (2001). Multi-scale non-linear FE^2 analysis of composite structures: damage and fiber size effects. *Revue européenne des Éléments Finis: NUMDAM'00 issue*, 10:449–472. (Cited on page 88.)

[Feyel et al., 1997] Feyel, F., Kruch, S., Roux, F.-X., and Cailletaud, G. (1997). Application of parallel computing to models with large numbers of internal variables. In $3^{ème}$ *Colloque National en Calcul de Structure*, number 83, pages 309–314, Giens, France. (Cited on page 63.)

[Fochesato et al., 2007] Fochesato, C., Grilli, S., and Dias, F. (2007). Numerical modeling of extreme rogue waves generated by directional energy focusing. *Wave Motion*, 44:395–416. (Cited on page 116.)

[Foerch, 1996] Foerch, R. (1996). *Un environement orienté objet pour la modélisation numérique des matériaux en calcul de structures*. Ph.D. Thesis, École Nationale Supérieure des Mines de Paris, France. (Cited on page 87.)

[Förster, 2007] Förster, C. (2007). *Robust methods for fluid-structure interaction with stabilised finite elements*. Ph.D. Thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany. (Cited on page 47.)

[Förster et al., 2007] Förster, C., Ramm, W. A., and Ramm, E. (2007). Artificial added mass instabilities in sequential staggered coupling of nonlinear structures and incompressible viscous flows. *Computer Methods in Applied Mechanics and Engineering*, 196:1278–1291. (Cited on pages 47, 99, 102 and 103.)

[Förster et al., 2006] Förster, C., Wall, W. A., and Ramm, E. (2006). On the geometric conservation law in transient ow calculations on deforming domains. *International Journal for Numerical Methods in Fluid*, 50:1369–1379. (Cited on pages 25, 45 and 102.)

[Fosdick et al., 1996] Fosdick, L. D., Jessup, E. R., Schauble, C. J. C., and Domik, G. (1996). *An introduction to high-performance scientific computing*. The MIT Press. (Cited on page 63.)

[Franca et al., 1993] Franca, L. P., Hughes, T. J. R., and Stenberg, R. (1993). Stabilized finite element methods. *Incompressible Computational Fluid Dynamics*, pages 87–107. (Cited on pages 7 and 37.)

[Frandsen, 2004] Frandsen, J. B. (2004). Numerical bridge deck studies using finite elements. part i: flutter. *Journal of Fluids and Structures*, 19(2):171–191. (Cited on page 1.)

[Fries, 2005] Fries, T.-P. (2005). *A Stabilized and Coupled MeshfreeMeshbased Method for Fluid-Structure Interaction Problems*. Ph.D. Thesis, Technischen Universität Carolo-Wilhelmina zu Braunschweig, Germany. (Cited on pages 27, 89 and 116.)

[Garaud, 2008] Garaud, J.-D. (2008). *Développement de Méthodes de couplage aéro-thermo-mécanique pour la prédiction des instabilités dans les structures aérospatiales chaudes*. Thèse de Doctorat, Université Pierre et Marie Curie, France. (Cited on page 87.)

[Geist, 2007] Geist, a. (2007). Pvm official home page. http://www.csm.ornl.gov/pvm. (Cited on page 65.)

[Gerbeau and Vidrascu, 2003a] Gerbeau, J. and Vidrascu, M. (2003a). A quasi-Newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows. Rapport de Recherche 4691, Institut National de Recherche en Informatique et Automatique, Rocquencourt, France. (Cited on page 40.)

[Gerbeau and Vidrascu, 2003b] Gerbeau, J. and Vidrascu, M. (2003b). A quasi-Newton algorithm based on a reduced model for fluid-structure interaction problems in blood flows. *Mathematical Modelling and Numerical Analysis*, 37(4):631–647. (Cited on pages 58, 98, 100 and 102.)

[Germain and Muller, 1990] Germain, P. and Muller, P. (1990). Introduction à la mécanique des milieux continus. $2^{\text{ème}}$ édition. (Cited on page 2.)

[Ghia et al., 1982] Ghia, U., Ghia, K. N., and Shin, C. T. (1982). High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *Journal of Computational Physics*, 48:387–411. (Cited on pages 96 and 97.)

[Ghidaglia et al., 2001] Ghidaglia, J.-M., Kumbaro, A., and Le Coq, G. (2001). On the numerical solution to two fluid models via a cell centered finite volume method. *European Journal of Mechanics/B Fluids*, 20(6):841–867. (Cited on page 27.)

[Ghidaglia and Pascal, 2005] Ghidaglia, J.-M. and Pascal, F. (2005). The normal flux method at the boundary for multidimensional finite volume approximations in cfd. *European Journal of Mechanics/B Fluids*, 24(1):1–17. (Cited on pages 14 and 24.)

[Glowinski et al., 2003] Glowinski, R., Ciarlet, P., and Lions, J. (2003). Numerical methods for fluids (part 3). *Handbook of numerical analysis*, 9(3). (Cited on page 13.)

[Golub and van Loan, 1996] Golub, G. H. and van Loan, C. F. (1996). *Matrix computations*. The Johns Hopkins University Press, Baltimore, London. (Cited on page 51.)

[Graff, 1975] Graff, K. (1975). *Wave Motions in Elastic Solids*. Oxford Press. (Cited on page 12.)

[Gropp et al., 1994] Gropp, W., Lusk, E., and Skjellum, A. (1994). *Using MPI: Portable Parallel Programming with the Message-Passing Interface*. MIT Press. (Cited on page 65.)

[Guyon et al., 1999] Guyon, M., Madec, G., Roux, F.-X., and Imbard, M. (1999). A parallel ocean model for high resolution studies. In *Proceedings of the 5th International Euro-Par Conference on Parallel Processing*, pages 603–607. Springer-Verlag London, UK. (Cited on page 63.)

[Hautefeuille, 2009] Hautefeuille, M. (2009). *Numerical Modeling strategy for Heterogeneous Materials: A FE Multi-scale and Component-based Approach*. Ph.D. Thesis, Université Technologique de Compiègne, Technische Universität Braunschweig and École Normale Supérieure de Cachan, France and Germany. (Cited on pages 38, 79, 88, 91 and 128.)

[Heil, 2004] Heil, M. (2004). An efficient solver for the fully coupled solution of large-displacement fluid–structure interaction problems. *Computer Methods in Applied Mechanics and Engineering*, 193(1-2):1–23. (Cited on page 56.)

[Heywood et al., 1996] Heywood, J., Rannacher, R., and Turek, S. (1996). Artificial boundaries and flux and pressure conditions for the incompressible Navier-Stokes equations. *International Journal for Numerical Methods in Fluid*, 22(5):325–352. (Cited on page 12.)

[Hilber et al., 1977] Hilber, H. M., Hughes, T. J. R., and Taylor, R. L. (1977). Improved numerical dissipation for time integration algorithms in structural dynamics. *Earthquake Engineering & Structural Dynamics*, 5(3). (Cited on page 11.)

[Hirt et al., 1997] Hirt, C. W., Amsden, A. A., and Cook, J. L. (1997). An arbitrary Lagrangian-Eulerian computing method for all flow speeds. *Journal of Computational Physics*, 135(2):203–216. (Cited on pages 2 and 8.)

[Hoffman and Johnson, 2007] Hoffman, J. and Johnson, C. (2007). Computational turbulent incompressible flow. (Cited on page 7.)

[Hortmann et al., 1990] Hortmann, M., Perić, M., and Scheuerer, G. (1990). Finite volume multigrid prediction of laminar natural convection: Bench-mark solutions. *International Journal for Numerical Methods in Fluid*, 11:189–207. (Cited on pages 96 and 97.)

[Hübner et al., 2004] Hübner, B., Walhorn, E., and Dinkler, D. (2004). A monolithic approach to fluid-structure interaction using space-time finite elements. *Computer Methods in Applied Mechanics and Engineering*, 193:2087–2014. (Cited on pages 37, 95, 106, 109 and 117.)

[Hughes et al., 1979] Hughes, T. J. R., Pister, K. S., and Taylor, R. L. (1979). Implicit-explicit finite elements in nonlinear transient analysis. *Computer Methods in Applied Mechanics and Engineering*, 17:159–182. (Cited on page 37.)

[Ibrahimbegović, 2006] Ibrahimbegović, A. (2006). *Mécanique non linéaire des solides déformables : Formulation théorique et résolution numérique par éléments finis*. Hermès Sciences – Lavoisier, Paris. (Cited on page 9.)

[Ibrahimbegović, 2009] Ibrahimbegović, A. (2009). *Nonlinear solid mechanics: Theoretical formulations and finite element solution methods*. Springer. (Cited on pages 2 and 7.)

[Ibrahimbegović and Mamouri, 2002] Ibrahimbegović, A. and Mamouri, S. (2002). Energy conserving & decaying implicit time-stepping scheme for nonlinear dynamics of three-dimensional beams undergoing finite rotations. *Computer Methods in Applied Mechanics and Engineering*, 191:4241–4258. (Cited on page 46.)

[Ibrahimbegović and Markovič, 2003] Ibrahimbegović, A. and Markovič, D. (2003). Strong coupling methods in multi-phase and multi-scale modeling of inelastic behavior of heterogeneous structures. *Computer Methods in Applied Mechanics and Engineering*, 192:3089–3107. (Cited on page 88.)

[Idelsohn et al., 2003] Idelsohn, S., Onate, E., and Del Pin, F. (2003). A lagrangian meshless finite element method applied to fluid-structure interaction problems. *Computers and Structures*, 81(8-11):655–671. (Cited on pages 2 and 116.)

[Idelsohn et al., 2004] Idelsohn, S. R., Onate, E., and Del Pin, F. (2004). The particle finite element method: a powerful tool to solve incompressible flows with free-surfaces and breaking waves. *International Journal for Numerical Methods in Engineering*, 61:964–989. (Cited on page 27.)

[Irons and Tuck, 1969] Irons, B. M. and Tuck, R. C. (1969). A version of the aitken accelerator for computer iteration. *International Journal for Numerical Methods in Engineering*, 1(3):275–277. (Cited on page 53.)

[Jasak, 1996] Jasak, H. (1996). *Error Analysis and Estimation for the Finite Volume Method with Applications to Fluid Flows*. Ph.D. Thesis, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine, London, G.-B. (Cited on pages 7, 16, 52, 73 and 75.)

[Jasak and Tuković, 2007] Jasak, H. and Tuković, Z. (2007). Automatic mesh motion for the unstructured finite volume method. *Transactions of FAMENA*, 30(2):1–18. (Cited on pages 13, 22, 24 and 72.)

[Joosten et al., 2009] Joosten, M. M., Dettmer, W. G., and Perić, D. (2009). Analysis of the block gauss-seidel solution procedure for a strongly coupled model problem with reference to fluid-structure interaction. *International Journal for Numerical Methods in Engineering*, 78(7). (Cited on page 51.)

[Jürgens, 2009] Jürgens, D. (2009). Survey on software engineering for scientific applications. Informatikbericht, Institute for Scientific Computing, Braunschweig, Germany. (Cited on pages 35, 64 and 91.)

[Karypis and Kumar, 1998] Karypis, G. and Kumar, V. (1998). *METIS, A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices*. University of Minnesota, Department of Computer Science, Minneapolis, MN, USA. http://www.cs.umn.edu/~karypis. (Cited on page 81.)

[Kassiotis, 2008] Kassiotis, C. (2008). Which strategy to move the mesh in the Computational Fluid Dynamic code OpenFOAM? Technical report, WiRe / LMT–Cachan, Germany / France. http://perso.crans.org/kassiotis/openfoam/movingmesh.pdf. (Cited on page 22.)

[Kassiotis et al., 2009a] Kassiotis, C., Colliat, J.-B., Ibrahimbegović, A., and Matthies, H. G. (2009a). Multiscale in time and stability analysis of operator split solution procedure applied to thermomechanical problems. *Engineering Computations*, 1-2:205–223. (Cited on pages 37 and 131.)

[Kassiotis and Hautefeuille, 2008] Kassiotis, C. and Hautefeuille, M. (2008). *coFeap's Manual*. LMT-Cachan, École Normale Supérieure de Cachan, Cachan, France. http://www.lmt.ens-cachan.fr/cofeap. (Cited on page 66.)

[Kassiotis et al., 2008] Kassiotis, C., Ibrahimbegović, A., and Matthies, H. G. (2008). Model tsunami on coastal protections by a multi-scale partitioned strategy. In Schrefler, B. A. and Perego, U., editors, *8th World Concress on Computational Mechanics (WCCM8) 5th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS)*, Venice, Italy. (Cited on pages 116 and 128.)

[Kassiotis et al., 2009b] Kassiotis, C., Ibrahimbegović, A., Matthies, H. G., and Brank, B. (2009b). Stable splitting scheme for general form of associated plasticity including different scales of space and time. *Computer Methods in Applied Mechanics and Engineering*, In Press, Corrected Proof. (Cited on page 131.)

[Kitware Inc., 2009] Kitware Inc. (2008–2009). Paraview home page. http://www.paraview.org/. (Cited on page 68.)

[Kohl and Bernholdt, 2002] Kohl, J. A. and Bernholdt, D. E. (2002). CCA home page. http://www.csm.ornl.gov/cca/. (Cited on page 66.)

[Krosche, 2009] Krosche, M. (2009). Ofoam's manual. Informatikbericht, Institute for Scientific Computing, Braunschweig, Germany. (In Preparation). (Cited on page 75.)

[Küttler et al., 2006] Küttler, U., Förster, C., and Wall, W. A. (2006). A solution for the incompressibility dilemma in partitioned fluid-structure interaction with pure Dirichlet fluid domains. *Computational Mechanics*, 38:417–429. (Cited on pages 97 and 98.)

[Küttler and Wall, 2008] Küttler, U. and Wall, W. A. (2008). Fixed-point fluid-structure interaction solvers with dynamic relaxation. *Computational Mechanics*, 43(1):61–72. (Cited on pages 3, 53, 54, 58 and 101.)

[Lawlor and Zheng, 1999] Lawlor, O. S. and Zheng, G. (1999). Charm++ home page. http://charm.cs.uiuc.edu/research/charm/. (Cited on page 66.)

[Le Tallec and Mouro, 2001] Le Tallec, P. and Mouro, J. (2001). Fluid structure interaction with large structural displacements. *Computer Methods in Applied Mechanics and Engineering*, 190(24-25):3039–3067. (Cited on pages 40 and 47.)

[Lesoinne and Farhat, 1996] Lesoinne, M. and Farhat, C. (1996). Geometric conservation laws for flow problems with moving boundaries and deformable meshes, and their impact on aeroelastic computations. *Computer Methods in Applied Mechanics and Engineering*, 134(1-2):71–90. (Cited on page 45.)

[Mac Ilroy, 1968] Mac Ilroy, M. D. (1968). Mass produced software components. NATO Software engineering conference, Garmish, Germany. (Cited on page 64.)

[Matthies et al., 2006] Matthies, H. G., Niekamp, R., and Steindorf, J. (2006). Algorithms for strong coupling procedures. *Computer Methods in Applied Mechanics and Engineering*, 195:2028–2049. (Cited on pages 48 and 58.)

[Matthies and Steindorf, 2002] Matthies, H. G. and Steindorf, J. (2002). Strong coupling methods. Informatikbericht, Institut für Wissenschaftliches Rechnen, Germany. (Cited on page 131.)

[Matthies and Steindorf, 2003] Matthies, H. G. and Steindorf, J. (2003). Partitioned strong coupling algorithms for fluid-structure interaction. *Computers and Structures*, 81:805–812. (Cited on pages 40, 58 and 95.)

[Mehl et al., 2008] Mehl, M., Brenk, M., Bungartz, H. J., Daubner, K., Muntean, I., and Neckel, T. (2008). An Eulerian approach for partitioned fluid-structure simulations on Cartesian grids. *Computational Mechanics*, 43(1):115–124. (Cited on pages 2 and 37.)

[Mok, 2001] Mok, D. P. (2001). *Partitionierte Lösungsansätze in der Strukturdynamik und der Fluid-Struktur-Interaktion*. Ph.D. Thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany. (Cited on pages 47, 53, 95 and 98.)

[Niekamp, 2005a] Niekamp, R. (2005a). *CTL Manual for Linux/Unix for the usage with C++*. Institut fur Wissenschaftliches Rechnen – TU Braunschweig. http://www.wire.tu-bs.de/forschung/projekte/ctl/files/manual.pdf. (Cited on page 79.)

[Niekamp, 2005b] Niekamp, R. (2005b). Software component architecture. Lecture note, Institut für Wissenschaftliches Rechnen, Germany. http://www.wire.tu-bs.de/forschung/projekte/ctl/files/ctl_course.pdf. (Cited on pages 3, 64, 66, 79 and 127.)

[Niekamp et al., 2009] Niekamp, R., Markovič, D., Ibrahimbegović, A., Matthies, H. G., and Taylor, R. L. (2009). Multi-scale modelling of heterogeneous structures with inelastic constitutive behavior: Part II–software coupling implementation aspects. *Engineering Computations*, 26:6–28. (Cited on pages 66 and 68.)

[Niekamp and Stein, 2002] Niekamp, R. and Stein, E. (2002). An object-oriented approach for parallel two- and three-dimensional adaptive finite element computations. *Computers and structures*, 80:317–328. (Cited on pages 64, 66 and 67.)

[Nobile, 2001] Nobile, F. (2001). *Numerical Analysis of Axisymmetric Flows and Methods for Fluid-Structure Interaction Arising in Blood Flow Simulation*. Ph.D. Thesis, École Polytechnique Fédérale de Lausanne, Swiss. (Cited on page 1.)

[OpenCFD LTD, 2009] OpenCFD LTD (2000–2009). Openfoam home page. http://www.opencfd.co.uk/openfoam. (Cited on pages 29, 35, 73 and 96.)

[O'Regan, 2008] O'Regan, G. (2008). *A Brief History of Computing*. Springer-Verlag, London, Great-Britain. (Cited on page 64.)

[Orenstein, 2000] Orenstein, D. (2000). Quickstudy: Application Programming Interface (API). *http://www.computerworld.com*. (Cited on page 65.)

[Park et al., 1997] Park, K. C., Justino, Jr, M. R., and Felippa, C. A. (1997). An algebraically partitioned FETI method for parallel structural analysis: algorithm description. *International Journal for Numerical Methods in Engineering*, 40:2717–2737. (Cited on page 38.)

[Pascal and Ghidaglia, 2001] Pascal, F. and Ghidaglia, J.-M. (2001). Footbridges between finite volumes and finite elements with applications to CFD. *International Journal for Numerical Methods in Fluid*, 37:951–986. (Cited on page 7.)

[Patankar, 1980] Patankar, S. V. (1980). *Numerical heat transfer and fluid flow*. Hemisphere Publishing Corporation, Washington, DC. (Cited on pages 13, 15 and 52.)

[Perić et al., 2006] Perić, D., Dettmer, W. G., and Saksono, P. H. (2006). Modelling fluid-induced structural vibrations: reducing the structural risk for stormywinds. In Ibrahimbegović, A., editor, *NATO Advanced Research Workshop*, ARW 981641, pages 239–268, Opatija, Croatia. (Cited on pages 37 and 106.)

[Piperno, 1995] Piperno, S. (1995). *Simulation numérique de phénomènes d'interaction fluide-structure*. Thèse de Doctorat, École Nationale des Ponts et Chaussées. (Cited on page 43.)

[Piperno, 1998] Piperno, S. (1998). Numerical simulation of aeroelastic instabilities of elementary bridge decks. Rapport de Recherche 3549, Institut National de Recherche en Informatique et Automatique, Sophia-Antipolis, France. (Cited on page 1.)

[Piperno, 2000] Piperno, S. (2000). *Contribution à l'étude mathématique et à la simulation numérique de phénomènes d'interaction fluide-structure*. Habilitation à Diriger les Recherches, Université Paris 6. (Cited on pages 14 and 42.)

[Piperno and Farhat, 1997] Piperno, S. and Farhat, C. (1997). Design and evaluation of staggered partitioned procedures for fluid-structure interaction simulations. Rapport de Recherche 3241, Institut National de Recherche en Informatique et Automatique, Sophia-Antipolis, France. (Cited on page 43.)

[Piperno and Farhat, 2001] Piperno, S. and Farhat, C. (2001). Partitioned procedures for the transient solution of coupled aeroelastic problems–Part II: energy transfer analysis and three-dimensional applications. *Computer Methods in Applied Mechanics and Engineering*, 190:3147–3170. (Cited on pages 43 and 46.)

[Roshko, 1952] Roshko, A. (1952). *Of the Development of Turbulent Wakes from Vortex Streets*. Ph.D. Thesis, California Institute of Technology, Pasadena, California. (Cited on page 106.)

[Ross et al., 2009] Ross, M. R., Sprague, M. A., Felippa, C. A., and Park, K. C. (2009). Treatment of acoustic fluid-structure interaction by localized Lagrange multipliers and comparison to alternative interface-coupling methods.

Computer Methods in Applied Mechanics and Engineering, 198(9-12):986–1005. (Cited on page 38.)

[Rusche, 2002] Rusche, H. (2002). *Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions*. Ph.D. Thesis, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine, London, G.-B. (Cited on pages 27 and 29.)

[Salençon, 2005] Salençon, J. (2005). *Mécanique des milieux continus*, volume Tome 1: Concepts généraux. Ecole Polytechnique, Palaiseau. (Cited on pages 7 and 36.)

[Schäfer and Turek, 1996] Schäfer, M. and Turek, S. (1996). Benchmark computations of laminar flow around a cylinder. *Notes on numerical fluid mechanics*, 52:547–566. (Cited on pages 3, 7, 16, 18, 20 and 106.)

[Schäling, 2009] Schäling, B. (2009). Die Boost C++ Bibliotheken. http://www.highscore.de/cpp/boost/. (Cited on page 72.)

[Slone et al., 2003] Slone, A. K., Bailey, C., and Cross, M. (2003). Dynamic solid mechanics using finite volume methods. *Applied Mathematical Modelling*, 27(2):69–87. (Cited on page 7.)

[Sobey, 1998] Sobey, R. J. (1998). Linear and Nonlinear Wave Theory. Lecture note, Leichtweiß-Institut für Wasserbau, Braunschweig. (Cited on page 26.)

[Srisupattarawanit et al., 2006] Srisupattarawanit, T., Niekamp, R., and Matthies, H. G. (2006). Simulation of nonlinear random finite depth waves coupled with an elastic structure. *Computer Methods in Applied Mechanics and Engineering*, 195:3072–3086. (Cited on page 116.)

[Steindorf, 2002] Steindorf, J. (2002). *Partitionierte Verfahren für Probleme der Fluid-Struktur Wechselwirkung*. Ph.D. Thesis, Technischen Universität Carolo-Wilhelmina zu Braunschweig, Germany. (Cited on pages 106, 109, 111 and 131.)

[Stoker, 1992] Stoker, J. J. (1992). *Water Waves: The Mathematical Theory and Applications*. Wiley-Interscience, New-York. (Cited on pages 26 and 29.)

[Stroustrup, 1986] Stroustrup, B. (1986). *The C++ Programming Language*. Addison-Wesley and ACM Press, Reading, Massachusetts. (Cited on page 67.)

[Szyperski, 1998] Szyperski, C. (1998). *Component Software – Beyond Object-Oriented Programming*. Addison-Wesley and ACM Press, Reading, Massachusetts. (Cited on page 64.)

[Taylor, 2008] Taylor, R. L. (March 2008). *FEAP – A Finite Element Analysis Program Version 8.2 User Manual*. Departement of Civil and Environemental Engineering – University of California at Berkeley. http://www.ce.berkeley.edu/~rlt/feap/manual.pdf. (Cited on pages 35 and 64.)

[Turek and Hron, 2006] Turek, S. and Hron, J. (2006). Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. *Lecture Notes in Computational Science and Engineering*, 53:371. (Cited on page 106.)

[Ubbink, 1997] Ubbink, O. (1997). *Numerical prediction of two Fluid Systems with sharp interfaces.* Ph.D. Thesis, Department of Mechanical Engineering, Imperial College of Science, Technology and Medicine, London, G.-B. (Cited on pages 27 and 29.)

[von Scheven, 2009] von Scheven, M. (2009). *Effiziente Algorithmen für die Fluid-Struktur-Wechselwirkung.* Ph.D. Thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany. (Cited on pages 95, 112, 113, 114 and 115.)

[Walhorn, 2002] Walhorn, E. (2002). *Ein simultanes Berechnungsverfahren für Fluid-Struktur-Wechselwirkungen mit finiten Raum-Zeit-Elementen.* Ph.D. Thesis, Fachbereich für Bauingenieurwesen, Technischen Universität Carolo-Wilhelmina zu Braunschweig, Germany. (Cited on pages 37, 95 and 117.)

[Walhorn et al., 2005] Walhorn, E., Kölke, A., Hübner, B., and Dinkler, D. (2005). Fluid-structure coupling within a monolithic model involving free surface flows. *Computers and Structures*, 83(25-26):2100–2111. (Cited on pages 3 and 117.)

[Wall, 1999] Wall, W. A. (1999). *Fluid-Struktur Interaktion mit stabilisierten Finiten Elementen.* Ph.D. Thesis, Institut für Baustatik und Baudynamik, Universität Stuttgart, Germany. (Cited on pages 3, 40, 47, 53, 54, 95, 97, 100, 102, 109, 111 and 115.)

[Wall and Ramm, 1998] Wall, W. A. and Ramm, E. (1998). Fluid-structure interaction based upon a stabilized (ALE) finite element method. Sonderforschungsbereich 404, Institut für Baustatik und Baudynamik, Germany. (Cited on pages 95, 96, 98 and 106.)

[Wang et al., 2008] Wang, H., Chessa, J., Liu, W., and Belytschko, T. (2008). The immersed/fictitious element method for fluid-structure interaction: Volumetric consistency, compressibility and thin members. *International Journal for Numerical Methods in Engineering*, 74(1). (Cited on page 8.)

[Weller et al., 1998] Weller, H. G., Tabora, G., Jasak, H., and Fureby, C. (1998). A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631. (Cited on page 73.)

[Zienkiewicz and Taylor, 2001a] Zienkiewicz, O. C. and Taylor, R. L. (2001a). *The Finite Element Method, Fluid Mechanics*, volume 3. Butterworth Heinemann, Oxford, 5 edition. (Cited on pages 7 and 12.)

[Zienkiewicz and Taylor, 2001b] Zienkiewicz, O. C. and Taylor, R. L. (2001b). *The Finite Element Method, Solid Mechanics*, volume 2. Butterworth Heinemann, Oxford, 5 edition. (Cited on page 7.)

[Zienkiewicz and Taylor, 2001c] Zienkiewicz, O. C. and Taylor, R. L. (2001c). *The Finite Element Method, The Basis*, volume 1. Butterworth Heinemann, Oxford, 5 edition. (Cited on page 9.)