A new robust co-simulation approach for transient Fluid-Structure Interaction problems

Marie Gibert

N° d'ordre : 2022LYSEI071

THÈSE de DOCTORAT DE L'UNIVERSITÉ DE LYON

Opérée au sein de l'INSA de Lyon

Ecole Doctorale 162

MEGA

Spécialité de doctorat : Génie mécanique

Soutenue publiquement 22/07/2022, par :

Marie GIBERT

A new robust co-simulation approach for transient Fluid-Structure Interaction problems

Devant le jury composé de :

Thouraya Baranger	Professeure	Univ Lyon 1	Présidente
Régis Cottereau	Chargé de Recherche	Centrale Marseille	Rapporteur
Bing Tie	Chargée de Recherche	Centrale Paris	Rapporteure
Sébastien Roth	Professeur	UTBM	Examinateur
Anthony Gravouil	Professeur	INSA-Lyon	Directeur de thèse
Michael Brun	Professeur	Univ de Lorraine	Co-Directeur de thèse
Valéry Botton	Professeur	INSA-Lyon	Invité
Valéry Morgenthaler	Docteur	Ansys France	Invité

Département FEDORA – INSA Lyon - Ecoles Doctorales

SIGLE	ECOLE DOCTORALE	NOM ET COORDONNEES DU RESPONSABLE
CHIMIE	**CHIMIE DE LYON** **https://www.edchimie-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage secretariat@edchimie-lyon.fr	**M. Stéphane DANIELE** C2P2-CPE LYON-UMR 5265 Bâtiment F308, BP 2077 43 Boulevard du 11 novembre 1918 69616 Villeurbanne **directeur@edchimie-lyon.fr**
E.E.A.	**ÉLECTRONIQUE, ÉLECTROTECHNIQUE, AUTOMATIQUE** **https://edeea.universite-lyon.fr** Sec. : Stéphanie CAUVIN Bâtiment Direction INSA Lyon Tél : 04.72.43.71.70 secretariat.edeea@insa-lyon.fr	**M. Philippe DELACHARTRE** INSA LYON Laboratoire CREATIS Bâtiment Blaise Pascal, 7 avenue Jean Capelle 69621 Villeurbanne CEDEX Tél : 04.72.43.88.63 **philippe.delachartre@insa-lyon.fr**
E2M2	**ÉVOLUTION, ÉCOSYSTÈME, MICROBIOLOGIE, MODÉLISATION** **http://e2m2.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.e2m2@univ-lyon1.fr	**M. Philippe NORMAND** Université Claude Bernard Lyon 1 UMR 5557 Lab. d'Ecologie Microbienne Bâtiment Mendel 43, boulevard du 11 Novembre 1918 69 622 Villeurbanne CEDEX **philippe.normand@univ-lyon1.fr**
EDISS	**INTERDISCIPLINAIRE SCIENCES-SANTÉ** **http://ediss.universite-lyon.fr** Sec. : Sylvie ROBERJOT Bât. Atrium, UCB Lyon 1 Tél : 04.72.44.83.62 secretariat.ediss@univ-lyon1.fr	**Mme Sylvie RICARD-BLUM** Institut de Chimie et Biochimie Moléculaires et Supramoléculaires (ICBMS) - UMR 5246 CNRS - Université Lyon 1 Bâtiment Raulin - 2ème étage Nord 43 Boulevard du 11 novembre 1918 69622 Villeurbanne Cedex Tél : +33(0)4 72 44 82 32 **sylvie.ricard-blum@univ-lyon1.fr**
INFOMATHS	**INFORMATIQUE ET MATHÉMATIQUES** **http://edinfomaths.universite-lyon.fr** Sec. : Renée EL MELHEM Bât. Blaise PASCAL, 3e étage Tél : 04.72.43.80.46 infomaths@univ-lyon1.fr	**M. Hamamache KHEDDOUCI** Université Claude Bernard Lyon 1 Bât. Nautibus 43, Boulevard du 11 novembre 1918 69 622 Villeurbanne Cedex France Tél : 04.72.44.83.69 **hamamache.kheddouci@univ-lyon1.fr**
Matériaux	**MATÉRIAUX DE LYON** **http://ed34.universite-lyon.fr** Sec. : Yann DE ORDENANA Tél : 04.72.18.62.44 yann.de-ordenana@ec-lyon.fr	**M. Stéphane BENAYOUN** Ecole Centrale de Lyon Laboratoire LTDS 36 avenue Guy de Collongue 69134 Ecully CEDEX Tél : 04.72.18.64.37 **stephane.benayoun@ec-lyon.fr**
MEGA	**MÉCANIQUE, ÉNERGÉTIQUE, GÉNIE CIVIL, ACOUSTIQUE** **http://edmega.universite-lyon.fr** Sec. : Stéphanie CAUVIN Tél : 04.72.43.71.70 Bâtiment Direction INSA Lyon mega@insa-lyon.fr	**M. Jocelyn BONJOUR** INSA Lyon Laboratoire CETHIL Bâtiment Sadi-Carnot 9, rue de la Physique 69621 Villeurbanne CEDEX **jocelyn.bonjour@insa-lyon.fr**
ScSo	**ScSo*** **https://edsciencessociales.universite-lyon.fr** Sec. : Mélina FAVETON INSA : J.Y. TOUSSAINT Tél : 04.78.69.77.79 melina.faveton@univ-lyon2.fr	**M. Christian MONTES** Université Lumière Lyon 2 86 Rue Pasteur 69365 Lyon CEDEX 07 **christian.montes@univ-lyon2.fr**

*ScSo : Histoire, Géographie, Aménagement, Urbanisme, Archéologie, Science politique, Sociologie, Anthropologie

INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE LYON

Abstract

A new robust co-simulation approach for transient Fluid-Structure Interaction problems

Marie GIBERT

This work proposes a coupling method for transient fluid-structure interaction (FSI) problems, based on a monolithic formulation and solved by a co-simulation algorithm. The aim is to use existing solvers for fluid and solid simulations and ensure a conservative coupling, with quasi-optimal level of efficiency, robustness and accuracy.
The coupled problem's monolithic formulation uses a Schur's dual approach, to ensure the normal velocities continuity at the interface. Then, the simulation algorithm is based on an extension of the GC method for FSI problems, to couple heterogeneous methods of discretization, integration scheme and time scales. The fluid sub-domain is discretized by finite volumes method and an explicit Runge-Kutta scheme. The solid sub-domain is discretized by finite elements method and an implicit Newmark scheme. Each sub-domain is driven by its own time scale.
The proposed method is validated with academic test cases. Then, it is integrated into the coupling library preCICE in order to run FSI simulations using existing CFD and structural dynamics three-dimensional solvers. Thus, the FSI benchmarks of the forward step problem and perpendicular flap problem are solved using OpenFOAM and CalculiX.

Contents

List of Figures

Introduction

Due to the need of understanding complex phenomena, dynamic multi-physics simulation is an active research field for the three past decades. In particular, many studies have focused on the numerical simulation of Fluid Structure Interaction (FSI) problems, for its wide range of applications, including aerospace [42, 72], civil engineering [101, 58, 10] and bio-medical [87, 75]. In this kind of problems, a fluid, air or gas, interacts with a solid material, with whom they share a common boundary called interface. The interface is impacted at the same time by the fluid pressure and solid displacement, see Fig.1. In the vast majority of cases, there is no analytical solution to solve FSI problems. Moreover, laboratory experiments are limited in scope; for example large scale problems as in civil engineering, expensive material as in aerospace application and living tissue as for biomedical application. Thus, in order to investigate the behaviour generated by the interactions between a fluid region and a solid material, numerical simulations may be employed.

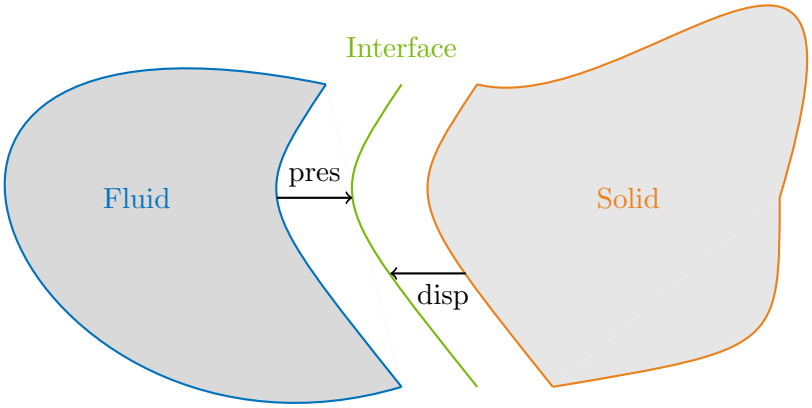


FIGURE 1: Interaction at the interface between a fluid sub-domain and a structural sub-domain

To solve FSI problems, two main approaches are usually used; partitioned method or monolithic method. In this work, they are defined according to their formulation. The formulation of the partitioned method is made of the formulation of each sub-domain, fluid and solid, in an uncoupled manner. Thus, the coupling condition of displacement and pressure are enforced as Dirichlet and Neumann boundary

conditions respectively. Partitioned methods have an appeal for their ease of implementation. Indeed, thanks to their uncoupled treatment of the FSI problem, this one can be solved using existing dedicated solvers over each sub-domain. Nevertheless, in the frame of transient dynamic resolution, to the staggered resolution, the resolutions of both sub-domains are not synchronous. This induced time-lag involves a loss of numerical accuracy from the interface coupling condition. Numerous improvements and complex partitioned methods have been developed in order, among other objectives, to limit the time-lag impact. Unfortunately, there is no proof of convergence in the general case using these approaches.
The second kind of FSI problems' formulations is called monolithic. Using these approaches, the global problem is solved synchronously, due to the fact that the fluid sub-domain, the structural sub-domain and the interface condition are written together in the same formulation. The advantages of these methods are that they are often more stable and more accurate than the partitioned approaches. Nevertheless, they are often based on unified algorithms of resolution that do not allow using an existing solver. Thus, the implementation of monolithic methods is difficult, more intrusive and not easily generalizable. That is why partitioned methods remain more popular than monolithic methods.

Then, ideally we would like to combine the strengths of both methods. First of all, the objective is consistency, convergence and stability between the exact, the discrete and the numeric solutions, as for any simulation, see Fig.2. To get closer to this goal, a coupling simulation based on a monolithic formulation seems to be more appropriate. Besides, it would be desirable for the computation to be efficient and easily usable with various cases. Therefore, partitioned approaches are really attractive. Thus, the idea of the proposed method is to extend the Schur's dual decomposition domain methods, as FETI methods developed for structural problems, to FSI problems. These approaches, in particular the GC method [53], are conservative and compute each sub-domain separately but synchronously.
Previous researches had shown promising results about the extension of the GC method to FSI problems. A fully explicit coupling of finite elements and finite volumes vertex centered has been proposed in [21]. Later, a multi-time scale coupling of SPH and finite elements has been developed [89]. More recently, a multi-time scale, fully lagrangian and explicit coupling of FE and PFEM, has been studied in [81].

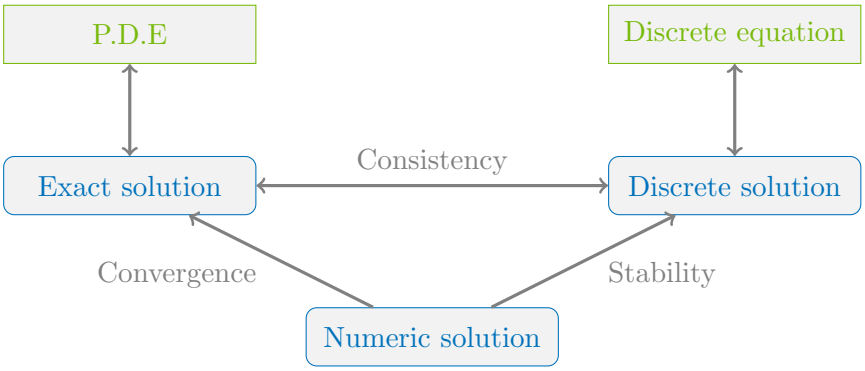


FIGURE 2: Relation between exact, numeric and discrete solutions

The main result of this thesis consists in an extension of the GC method to FSI problems, using: formulations, spatial discretization methods, specific time integrators and time/space scales over both sub-domains, very common and suited for each physic

(solid or fluid). In this manner, a co-simulation algorithm of resolution can be developed using efficient dedicated solvers over each sub-domain. Thus, the discretization methods chosen have to be available in commercial and free existing solvers. To compute the solid sub-domain, the choice was made to use the Finite Element Method (FEM) in Lagrangian formulation and an implicit Newmark integration scheme, as in Ansys Mechanical [6] or CalculiX [33]. Concerning the fluid sub-domain, the finite volumes cells centered in Arbitrary Lagrangian Eulerian (ALE) formulation is used with an explicit Runge-Kutta scheme.
Concerning the fluid-solid coupling, the dual problem is written as a monolithic formulation, where the coupling condition is the normal velocity continuity at the interface, enforced by the mean of Lagrange multipliers. Thus, the discrete problem can be solved by a co-simulation algorithm. Thanks to this, a dedicated solver is used for the sub-domains computation. More over, each sub-domain can use their own spatial and temporal scales.

The first chapter of this thesis is dedicated to an overview of FSI approaches, in order to justify the motivation of the proposed coupling method. A classification of FSI problems is proposed. Also, the main approaches of interface motion management are presented, as well as a short state of the art about domain decomposition methods.
Then the second chapter presents the method and its proof of concept. First, discretized equations solved over fluid and solid sub-domain are presented. Then the proposed coupling method is described in details from its monolithic formulation to its co-simulation algorithm of resolution. Finally, the method is validated thanks to the academical one-dimensional test case of the piston.
The last chapter, sets out the implementation of the proposed method using existing free software. The chosen solvers and coupling library are introduced. Afterwards, the implementation of the proposed method in the library preCICE is described. To finish, numerical results are presented thanks to the forward step and the perpendicular flap benchmarks.
Conclusions and perspectives are discussed into the last chapter.

Chapter 1

Overview of approaches for Fluid-Structure interaction problems

This first chapter presents a state of the art about FSI simulation. The objective here is to develop issues of this particular kind of simulation, where structural and fluid sub-domains are in interaction. Moreover, the aim is to present different approaches commonly used to solve FSI problems as well as their advantages and drawbacks.
The first section presents a classification of the FSI problems in terms of physical interactions and in terms of mathematical resolution. Then, special issues of the FSI problems due to the fluid-structure interface are discussed. Finally, a succinct state of the art of sub-domains decomposition methods is proposed, as well as a review of the extension of these methods for FSI problems.

1.1 Classification of FSI problems

The literature about FSI is vast and uses a rich and varied vocabulary. The difficulty is that there are no consensus or homogeneity about methods of classification and description of FSI problems. This is why in this work, we try to carefully define every term used.
The choice is to sort FSI problems according to two axes, see Fig.1.1. First we consider the coupling problem in terms of physics. In this scope, a coupling can be considered from weak, for example in the case of small deformations, to very strong, as needed for biomedical applications. In the other way, a coupling problem can be defined according to its numerical coupling. The following section proposes a state of the art of FSI simulation in accordance with these two aspects of coupling.

1.1.1 Physical coupling

Fluid-structure interaction problems cover a large field of application domains, where different kinds of physics step in. Two main categories of FSI applications can be differentiated, when the fluid is a gas and when the fluid is a liquid.

First, when the fluid is a gas, it is most often considered inviscid or weakly viscous and compressible. This is the case as example for the field of aerospace problems, which was one of the first active research fields for FSI problems. In [12] or [42] for

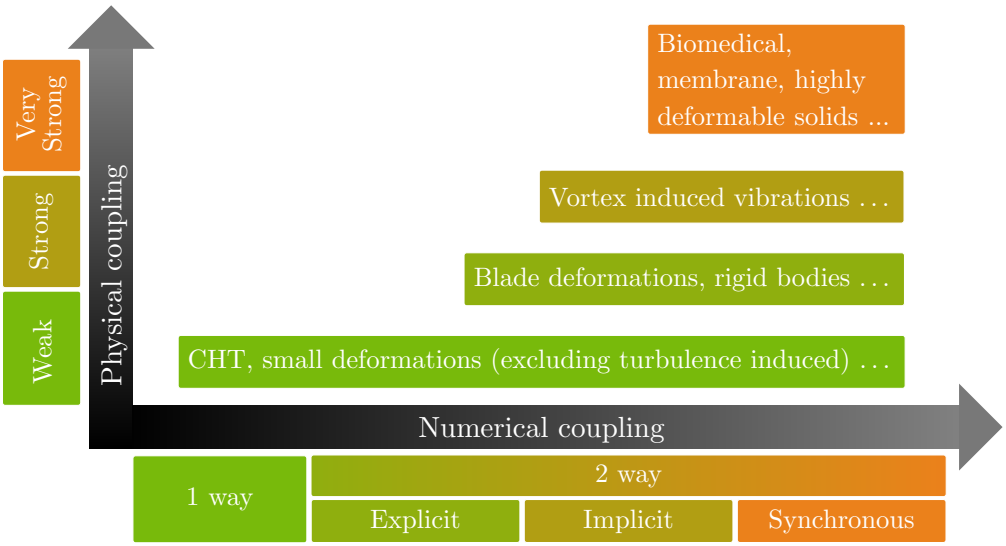


FIGURE 1.1: Coupling classification for FSI problems

example, authors study the fluid flow around deformable aircraft wing. The air flow around a rocket engine is also studied as in [72]. Recently, more complex simulation have been performed to evaluate the stability characteristic of a spacecraft parachute system [109].
The study of airflow around deformable structure is not only a major point of interest for flying objects but also for civil engineering. We can cite the well known example of the Tacoma Narrows bridge which has been destroyed in 1940 after large oscillation induced by aero-elasticity [69]. This event still remains an active subject of research for FSI, see [108]. In the civil engineering domain, we can also cite the study about flows around buildings. In [118], the authors investigated the turbulent flow past a hemispherical building. Finally, a really active research field these past years is the simulation of wind turbines as in [10] and [105] as example.
Concerning FSI simulation where the fluid is a gas, a really challenging field of research is about biomedical applications. The phonation can be studied using these FSI simulation models. In [75], the authors proposed a 2 dimensional Finite Element approach to perform fluid-solid acoustic simulation of the human phonation. The respiratory system is also studied thanks to the FSI simulation approaches. As example, [77] proposed a simulation FSI model of the trachea and [116] studied lower airways. In most of these studies, difficulties obviously come from the coupling problem but also from the fluid's turbulence. In another way, the solid sub-domain is often considered linear elastic for the objective is to characterize the instability rather than computing the post-breaking behaviour. The biomedical applications are though an exception for the considered solids are often non-linear in terms of geometry and material characteristics.

The other cases are when the fluid is liquid. Most of the time it is considered viscous and incompressible or nearly incompressible. In this range of studies, hydraulic application cases are numerous. There is the family of problems where a fluid flows around a network of thermal exchangers, as used in nuclear plants. A review of this kind of problems is proposed in [97]. There also is the group of the tanks problems, where the fluid is enclosed into a solid in motion. As examples, [5] studied compressibility and gravity effects in internal vibrations and [66] proposed to simulate a rocket tank. Finally, hydraulic problems also treat of the range of turbine problem, of which a

review is proposed in [112].
The FSI problem with liquid fluid is also a challenge for civil engineering. We can cite the study of dams, in their normal states as in [101] or in breaking [103], [32]. Submissive waves as tsunami are also studied in the FSI scope as in [58].
Finally, one of the most recent and dynamic field of research of liquid-solid coupled simulation is biomedical applications. The first researches have focused on blood flow circulation, and remains really active today [11], [14]. Then, hemodynamic fluid-structure coupling problems have been proposed for complex pathologies and patient specific models as cerebral aneurysms [111] or aorthica dissection [2] as examples. FSI simulation is also used for eye modeling. [62] studied the influence of intraocular pressures on the human eye and [61] proposed a model to compute injury to the eye caused by glass splinters.

This large field of application, deals with various kind of physics. Thus, according to the physics of fluid and solid, the influence on the FSI phenomenon of each sub-domain on the other is variable. If this influence is minor, the coupling is called weak and respectively strong when major. To measure this fluid-structure influence, two dimensionless numbers are commonly used. The first one is the mass number $\mathcal{M}$, which is the ratio of the fluid and solid density:

$$\mathcal{M} = \frac{\rho_s^0}{\rho_f^0} \tag{1.1}$$

Where ρ_s^0 is the initial structural density and ρ_f^0 is the initial fluid density. The more this ratio is close to 1, the stronger is the coupling. The other one is the Cauchy number $\mathcal{C}$, which measures the magnitude of dynamic-induced deformations.

$$\mathcal{C} = \frac{\rho_f^0 \underline{\mathbf{v}}_f^0 . \underline{\mathbf{v}}_f^0}{E_s} \tag{1.2}$$

Where $\underline{\mathbf{v}}_f^0$ is the initial fluid velocity and E_s is the Young modulus of the solid sub-domain. A weak number of Cauchy induces a weak physical coupling, with a very low density fluid and a rigid solid for example. Thus, a weak or a strong coupling is not induced by the fluid's state, being gas or liquid. From the previous examples, the coupling model of the parachute is weaker than the aircraft wing FSI simulation according to mass number and Cauchy number.
The objective while determining the physical coupling strength of the considered problem is to choose an appropriate numerical coupling method, to solve it. The stronger the physical coupling is, the more complex the numerical coupling should be.

1.1.2 Numerical coupling

Concerning the numerical coupling, a large variety has been investigated, leading to several classifications and definitions. Nevertheless, FSI problems are traditionally classified on two main categories: partitioned methods and monolithic methods. Partitioned coupling methods treat FSI problems in an uncoupled way. In the other way, with monolithic methods, the fluid sub-domain, the structural sub-domain and the interface between them are thought together. The main advantages of partitioned methods are their easy implementation, because existing CFD and structural dynamic

codes can be used successively. In this way, they are flexible and easily generalizable from an FSI problem to another. Unfortunately, those kind of numerical coupling methods lack accuracy due to the time lag, which is inherent to their formulation. The monolithic methods turn out to be better in terms of accuracy and stability, thanks to their synchronous resolution, but are often difficult to implement and not easily generalizable. Then they are often developed for specific applications.

In literature, FSI coupling problems are always divided into partitioned or mono-

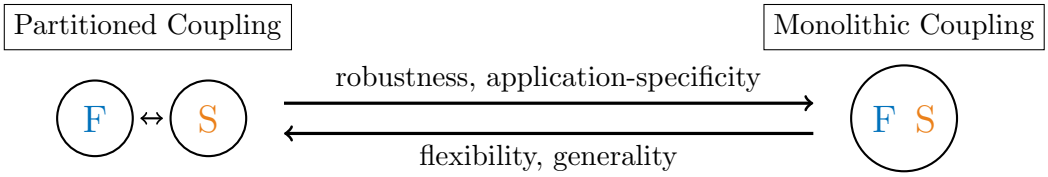


FIGURE 1.2: Numerical coupling approaches of FSI: partitioned or monolithic coupling

lithic coupling. Even if the general definition is most of time really close, and matches the definition given previously, there are some differences in the approach depending on the authors. Some authors, as De Boer in [29], consider that the difference between a partitioned coupling method and a monolithic coupling method is in term of solvers. A partitioned solution solves each sub-domain thanks to a dedicated solver, while a monolithic solution is mono-code. In the other way, some authors as Benra in [13], consider the FSI coupling classification in terms of formulation. For partitioned coupling, the formulation of the FSI problem is written as the continuous formulation of each sub-domain and then, coupling condition at the interface are added as boundaries condition. Meanwhile, a monolithic approach treats the fluid sub-domain, solid sub-domain and interface formulation globally to solve the coupled problem synchronously. Finally, authors also mixed these two approaches in literature as in [56]. We chose to follow the same approach as Benra and classify FSI problems as partitioned or monolithic coupling according to their formulations. Thus, in the following, numerical coupling methods are described and refer to the chosen formulation of FSI problems, which is neither because of their algorithms of resolution nor their kind of solvers used.

Partitioned coupling

Partitioned coupling methods have been the first proposed and have had a lot of improvements through the last decades. The following subsection proposes to describe coarsely the main kind of serial partitioned methods.

The easiest partitioned coupling is the one-way coupling. In this method, fluid and solid sub-domains are solved separately at each temporal iteration and only one physic impacts the other. Most of time, this coupling uses the hypothesis of rigid walls. The fluid sub-domain is computed alone, but the fluid pressure is used to solve the solid state, see Fig.1.3. This figure shows the procedure followed at each time step n to solve a one-way coupling. First, the fluid sub-domain is solved alone and some fields are known as solution state at the current time step, x_f^{n+1}, fluid pressure for example. Then, this pressure at boundaries between the fluid sub-domain and the solid sub-domain is used as Neumann boundary conditions to solve the structural state at t^{n+1}. This kind of coupling is well suited for weak physical coupling, where small deformations concerning the solid sub-domain occur, but the stress induced by the fluid pressure into the structure is reached as in [84]. This kind of coupling is also

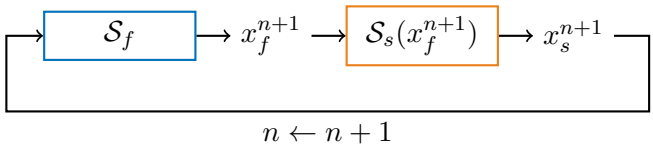


FIGURE 1.3: One-way partitioned serial explicit coupling where the first computation is the fluid sub domain

often used as a first approach of very strong coupling, as for biomedical application. In [63], aneurysm are studied considering vessels as rigid walls for the blood flow computation and the fluid pressure is used to compute stress induced into vessels. The reverse one-way coupling is also used, when the solid boundaries are moving but considered non deformed by the fluid pressure, as in this simulation of peristaltic pump [90]. These kinds of one way numerical-coupling are interesting due to their lower computational time cost in comparison with two-way couplings, but they are most of the time not close enough to real FSI problems.

To perform more accurate FSI simulations, two-way numerical couplings are used. With this method, both sub-domains are alternately integrated in time. The interaction is taken into account by the boundary conditions of both solvers. The first two-way coupling method has been proposed by Park *et al.* [91], here, we call it partitioned serial explicit coupling, see Fig.1.4. As previously mentioned, we consider that the fluid sub-domain is solved first. Thus, contrary to the one-way method, the fluid is not solved alone but using an output field from the previous time step solution of the solid x_s^n as an input: displacements or velocities of the boundaries for example. Then, using the fluid solution at the current time step x_f^{n+1} as boundary conditions, the solid sub-domain is solved and the temporal time step is increased. As a consequence there is a time lag between the integrations of both fluid and struc-

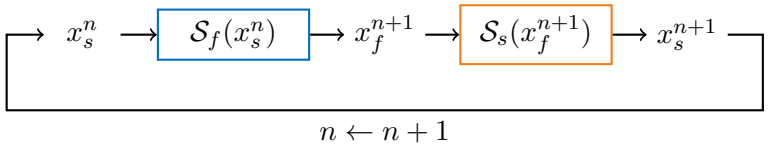


FIGURE 1.4: Two-way partitioned serial explicit coupling

tural sub-domains. Indeed, if the solid state is solved at the instant t^{n+1} using fluid variables computed during the current times step, it is not the case for the fluid state computation which uses solid variables from the previous time step. This kind of sequential coupling method can create numerical instability as added mass effect [18].

In order to stabilize the solution from two-way partitioned explicit coupling, a fixed-point iteration can be performed. This kind of coupling method is called serial implicit coupling and has been introduced by Alonso and Jomeson [3]. The procedure on each time step is the same as for the explicit serial coupling except that coupling iterations can occur between the fluid and the solid solution, until convergence (or until the maximum number of coupling sub-iterations is reached). Classical Gauss-Seidel method is often used as in [60]. Quasi-Newton methods are also useful for implicit FSI coupling as IQN-ILS [30] or MVQN [17].
Despite a large number of coupling iterations, the convergence is not guaranteed, due to the fact that there is no proof of stability concerning the staggered resolution of

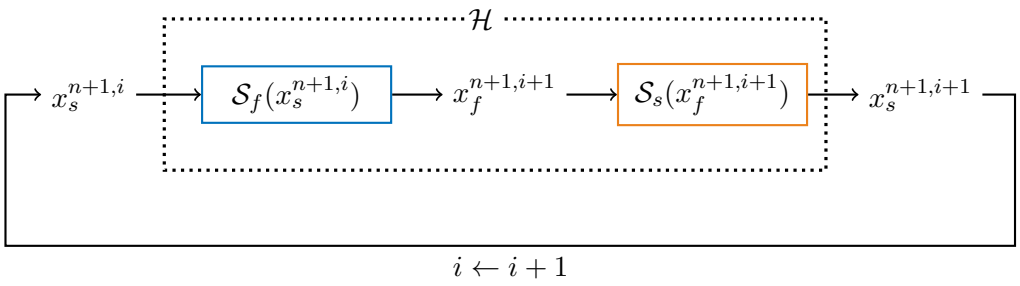


FIGURE 1.5: Two-way partitioned serial implicit coupling, after a non-convergence iteration, the latest stored state is reloaded while when the solution converges, the time step n is increased

the partitioned coupling in the general case [85].
Nevertheless, lots of researches have been proposed to improve the partitioned coupling approach. This short state of the art only describes serial partitioned coupling but explicit and implicit coupling methods have been parallelized [106]. Predictors of the fluid pressure or of the interface displacement are also used to minimize the mentioned time lag. Numerous other improvements have also been brought to the partitioned methods such as different time-steps, complex integration time schemes and discretization methods for fluid and structural domains.

Monolithic coupling

In this context, the monolithic numerical couplings propose a synchronous resolution instead of a sequential resolution of FSI problems. They were introduced later than the partitioned coupling method due to the fact that they are less intuitive. In 1998, Blom [16] has extended the work of Feldkler [43], concerning static problems solved by monolithic coupling, to dynamic FSI problems. He proposed a monolithic coupling for a one-dimension problem using classical discretization method for both sub-domains using the same time step. The formulation can be called unified, the entire FSI problem, fluid, solid and interface, being solved by a unique solver. The results are quite good in accuracy, but the generalization of such methods is difficult. Later, most of the proposed monolithic coupling used "structural" methods to compute the fluid sub-domain using a unified formulation, including fluid computation into structural solver [99], or completely re-building a new solver [9]. To improve monolithic methods, Ryzhakov *et al.* proposed a Lagrangian formulation of the fluid sub-domain and Mayr *et al.* [80] set up a coupling method where the temporal and the spatial scales are different for fluid and structure.
However, the main drawbacks of the monolithic coupling methods still remain, namely generalization. Indeed, in this kind of numerical coupling, the global formulation of the FSI problem is unified and uses a primal formulation of interface coupling. Then, particular solvers which are dependent on the hypothesis of the considered FSI problem are used. In this way, these numerical couplings are not often easily generalizable. To minimize this drawback, monolithic formulations based on dual coupling and decomposition methods have been proposed and will be presented later in section 1.3.2.

In this section, the variety of FSI problems in terms of application fields and in terms of methods of resolution has been presented. FSI problems can be classified according to their numerical coupling and according to their physical coupling. In this work, concerning the numerical couplings, the choice is made to describe them according to

their formulation. The proposed classification of the numerical coupling is summed up on Fig.1.6.

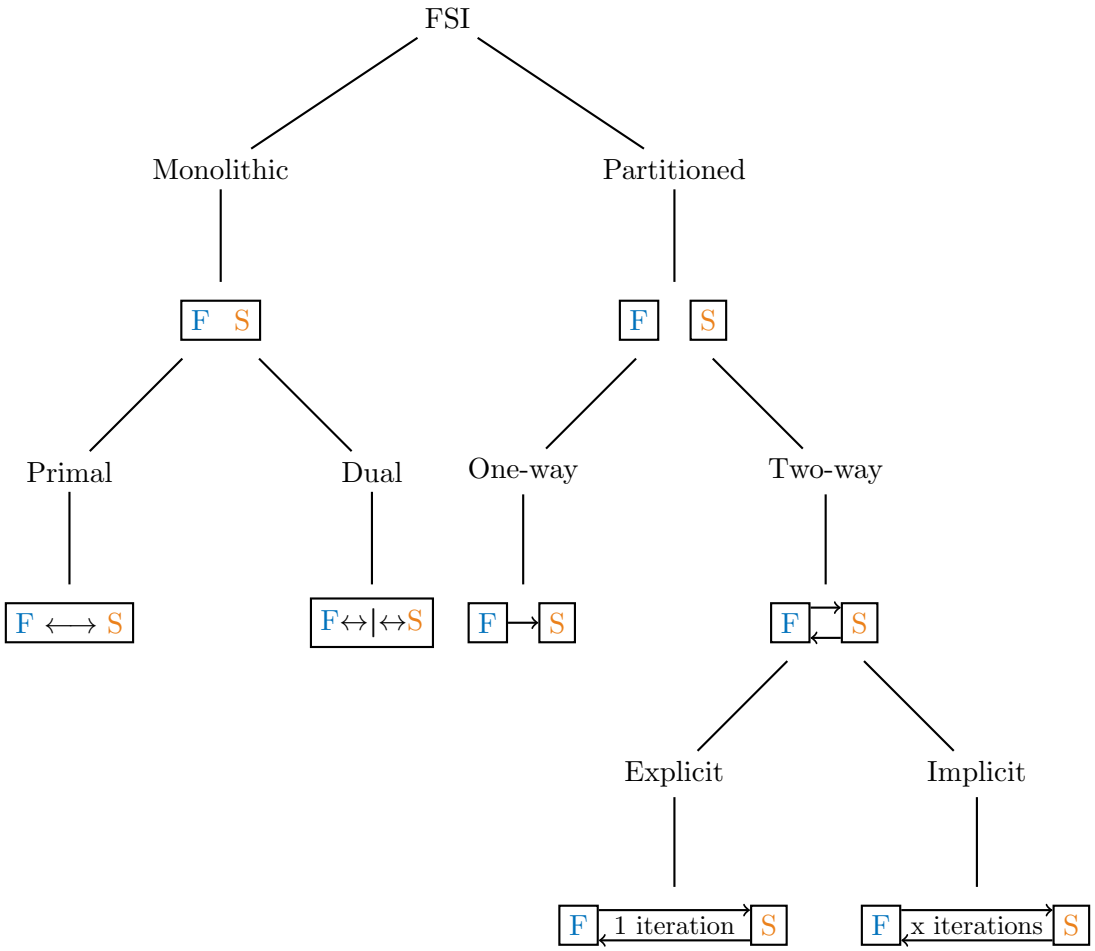


FIGURE 1.6: Proposed classification of FSI numerical coupling

1.2 Fluid-Structure interface specificity

As said in the previous section, application fields of FSI problems are very diversified. Then, multiple physics can step in, as well as various configurations. In this section, the hypotheses of considered FSI problems are described in terms of the studied physics and geometry. Moreover FSI problems have to deal with both physics and with the behaviour of the interface between fluid and structural sub-domains. This characteristic of coupling problems has a particular issue in the case of FSI due to the different approaches of referential historically used for CFD and structural dynamic problems.

1.2.1 Reference problem

In this work, the studied FSI problems considered are reduced to continuous mechanics. A fluid, considered inviscid and compressible, is in contact with a linear elastic solid. Free-surfaces problems are not treated, neither are porous materials. Those

hypotheses are chosen arbitrarily, to fit with the test case simulation used for the proposed method and presented in the next chapters. Nevertheless, one of the objectives of the proposed method being to be generalizable to a large case of FSI simulation, a discussion about the extension of physical hypotheses is presented in conclusion.

Geometry and sub domain definition

An enclosed FSI domain Ω is considered. Fig.1.7, shows a representation of studied FSI problems and notations used.

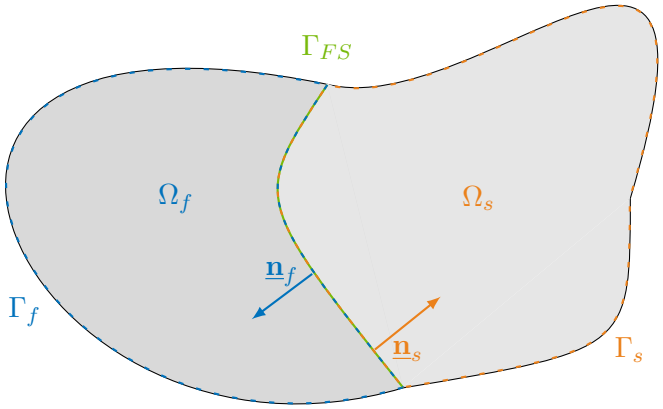


FIGURE 1.7: FSI problem geometry 2D representation

The solid sub-domain is called Ω_s, considered homogeneous, continuous and isotropic and is closed by the boundary Γ_s. Respectively, the fluid sub-domain is called Ω_f, considered continuous, compressible and inviscid, and is closed by the boundary Γ_f. There is no inter-penetration, overlap or empty space between the two sub-domains, $\Omega_s \cup \Omega_f = \Omega$ and $\Omega_s \cap \Omega_f = \emptyset$. The two sub-domains are in contact through the common part of their boundaries. This interface is called Γ_{FS}. The external normals of the interface, are called $\underline{\mathbf{n}}_s$ and $\underline{\mathbf{n}}_f$ for solid and fluid sub-domains respectively, such as $\underline{\mathbf{n}}_s = -\underline{\mathbf{n}}_f$.
The behaviour of the FSI problem is studied from the initial time $t^0 = 0$ to the final time of the simulation $t^f = T$. The state variables over the solid sub-domain, displacement $\underline{\mathbf{d}}_s$, velocity $\underline{\mathbf{v}}_s$ and acceleration $\underline{\mathbf{a}}_s$ are linked to Ω_s as well as the secondary variables stress $\underline{\underline{\sigma}}_s$ and strain $\underline{\underline{\epsilon}}_s$. Initial conditions and boundary conditions are additionally imposed. On Γ_s, Dirichlet condition imposes structural displacement $\underline{\mathbf{d}}_D$ while Neumann condition imposes external forces $\underline{\mathbf{f}}_{ext}$. Concerning the fluid sub-domain Ω_f, the conservative variables are considered, namely density ρ_f, momentum $\rho_f \underline{\mathbf{v}}_f$ and energy E_f. Then the secondary variables, temperature T_f and p_f are calculated. As for the solid sub-domain, initial and boundary conditions are added. On Γ_f fluid velocities or pressure are imposed.
Finally, coupling conditions on the interface Γ_{FS} have to be added to completely

define the fluid-structure problem. First, a kinetic condition is needed to enforce the fluid and the solid sub domain to always fit, no detachment or overlap can occur. Thus, the velocity continuities are enforced by equation (1.3a) Moreover, the solid being deformable, a dynamic condition is needed to impose that the stress in the fluid is equal to the stress in the solid, equation (1.3b).

$$\underline{\mathbf{v}}_s = \underline{\mathbf{v}}_f \qquad \text{on} \quad \Gamma_{FSI} \times [0,T] \tag{1.3a}$$

$$\underline{\underline{\sigma}}_s \cdot \underline{\mathbf{n}}_s + \underline{\underline{\sigma}}_f \cdot \underline{\mathbf{n}}_f = \underline{\mathbf{0}} \qquad \text{on} \quad \Gamma_{FSI} \times [0,T] \tag{1.3b}$$

Due to the inviscid hypothesis of the fluid, there is slip on Γ_{FS}. Thereby, the coupling condition (1.3) are re-written such as:

$$(\underline{\mathbf{n}}_s \cdot \underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s = (\underline{\mathbf{n}}_f \cdot \underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f \qquad \text{on} \quad \Gamma_{FSI} \times [0,T] \tag{1.4a}$$

$$(\underline{\underline{\sigma}}_s \cdot \underline{\mathbf{n}}_s)\underline{\mathbf{n}}_s = p_f \qquad \text{on} \quad \Gamma_{FSI} \times [0,T] \tag{1.4b}$$

Thus, the FSI problem considered is well defined.

Management of the mobile fluid-structure interface

As said previously, the main point of FSI problem is the management of the coupling conditions at the interface but also its motion. Indeed, due to the solid deformation and the fluid pressure, the boundary Γ_{FS} is mobile. For structural dynamic computation, mobile boundaries are not a particular problem, and are usually defined into the lagrangian referential, see section 1.2.2. Nevertheless, for CFD, the moving wall is actually an issue, due to the eulerian referential, historically used for fluid computation, see section 1.2.2. Thus, two kinds of approaches have been proposed in literature to treat the interface motion for the fluid computation; interface tracking and interface capturing, Fig. 1.8.

Interface capturing approach has been the first proposed in frame of FSI in [92] and a review of these methods can be found in [83] and [64]. With interface capturing, the fluid domain remains fixed according to time. Hence, a gap between the two sub-domains as well as a overlap can occur. Coupling conditions are imposed thanks to immersed boundary method as in [119], fictitious domain-mortar element method as in [8], chimera method as in [104], arlequin method as in [44] and so on.
These methods are really effective because classical CFD methods can be used and no re-meshing or mesh motion are needed. However, the interface position being estimated for the fluid computation, coupling conditions and exchanged fields at the interface can lose accuracy.

The other way to manage the fluid-structure interface motion is called interface tracking. Using this approach, the boundary Γ_{FS} is computed at each time step for both fluid and solid computations. Thus, the fluid sub-domain and the structural sub-domain always fit. Hence, to fit the interface Γ_{FS}, the fluid sub-domain has to move according to time. In order to achieve that, the fluid sub-domain can use a eulerian formulation and be re-meshed every time step but it will be incredibly costly. Then, deformable meshes are used instead over the fluid sub-domain. Lagrangian formulation can be used as for the solid computation, as proposed in [45], or ALE formulation

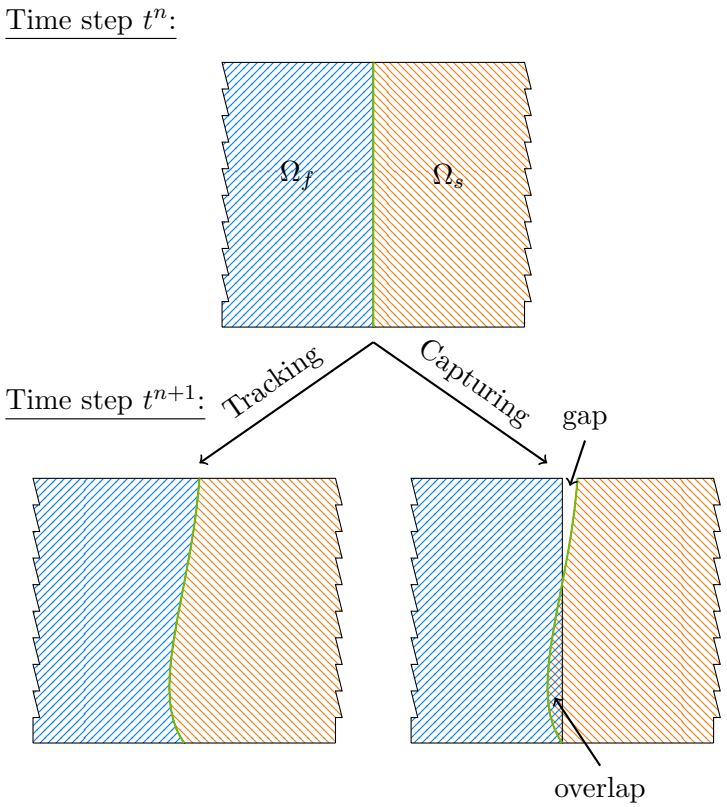


FIGURE 1.8: Management of the interface motion by tracking or capturing for the fluid sub-domain

(see section1.2.2) as [65].
This approach is more accurate than interface tracking but is often more costly in computation time. Indeed, if the interface displacements are large, the fluid sub-domain should be re-meshed to avoid poor quality elements, which is a costly step. Even using meshless methods, as SPH methods based on Lagrangian formulation in [107] to ensure the interface tracking, re-meshing can occur.

Regarding the aims of the proposed method, an interface tracking approach is used in chapter 2. Interface capturing approaches are more efficient but are less accurate concerning the interfaces values. The proposed method of coupling having the objective of ensuring no interface energy dissipation, an accurate method for the interface values seems to be more suitable.

1.2.2 Specific formulation of fluid and solid sub-domains

As said previously, issues concerning FSI problems is on one hand, the management of the mobile interface (by interface tracking in this work), and on the other hand, the coupling of two physics using historically different methods of computation including different formulations. In continuum mechanic frame, the main formulations are the Lagrangian formulation used for solid problem, and the Eulerian formulation used to describe fluid flow. These two formulations are described bellow as well as a third one, the Arbitrary Lagrangian Eulerian (ALE) formulation commonly used for FSI tracking in FSI problems.

Eulerian formulation

The eulerian formulation is based on the spatial domain Ω_x, the referential domain fixed in time. It is described by the spatial system of coordinate $\underline{\mathbf{x}}$. Then the physical variables associated to material particles are studied passing through a fixed region of the space.
At a discretized level, as shown on Fig.1.9, the mesh nodes (orange crosses) are where variables are computed and remain fixed between two time steps. As often, here the material domain is chosen to be the spatial domain at initial configuration. Then, the referential grid, mesh nodes and connection are not mobile. The material particles, blue circles, are moving between two time steps, and it's not the quantities at the materials point that are computed but the quantities of the particles crossing fixed grid elements.

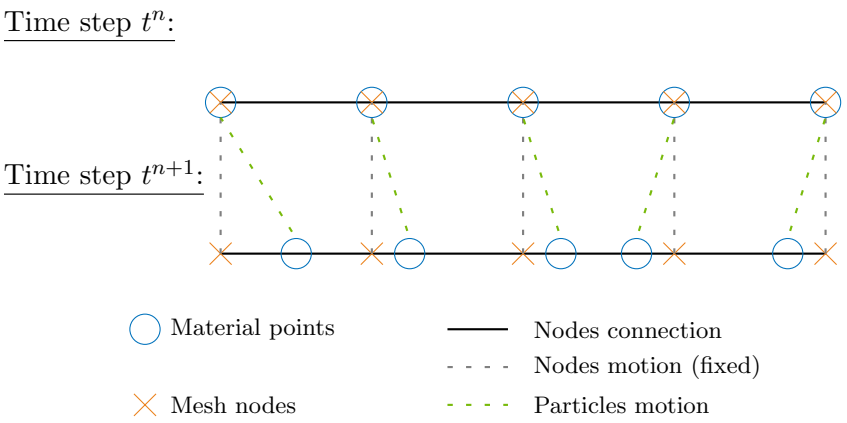


FIGURE 1.9: Eulerian mesh motion between two time-steps

In this way, the Eulerian formulation allows to compute complex material motion since mesh nodes are dissociated from the particles. Thus, this formulation is typically used in fluid mechanics computation. Using this formulation, the material's boundaries are not computed which is problematic when interface tracking is used.

Lagrangian formulation

The Lagrangian formulation is based on the material domain called Ω_X. The Lagrangian referential moves with the material domain deformation. The material coordinates are noted $\underline{\mathbf{X}}$. The mapping φ between the material domain and the spacial domain is defined as:

$$\begin{aligned} \varphi : \Omega_X \times [0,T] &\rightarrow \Omega_x \times [0,T] \\ (\underline{\mathbf{X}}, t) &\mapsto \varphi(\underline{\mathbf{X}}, t) = (\underline{\mathbf{x}}, t) \end{aligned} \tag{1.5}$$

We also define the material velocities $\underline{\mathbf{v}}$ as:

$$\underline{\mathbf{v}}(\underline{\mathbf{X}}, t) = \frac{\partial \underline{\mathbf{x}}}{\partial t}|_{\underline{\mathbf{X}}} \tag{1.6}$$

Fig.1.10 shows a representation of discretized Lagrangian formulation. For every time step, grid nodes are connected with the same material point. Nodes motions

and particles motions are combined. Physical variables associated with a particle are computed at its attached node.

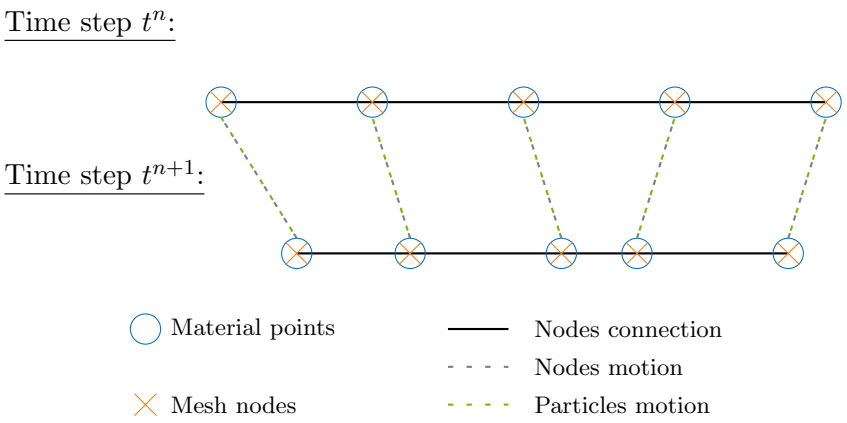


FIGURE 1.10: Lagrangian mesh motion between two time-steps

The Lagrangian frame then allows the tracking of moving boundaries. It is also useful to take into account material history to compute the domain behaviour. These two advantages are suitable for structural dynamic computation. For fluid flow, really large deformation can occur, with vortices for example, the particle and then the mesh motion can critically distort the grid.

ALE formulation

In the frame of fluid simulation for FSI problems with interface tracking, it appears that neither the eulerian formulation, classically used (which cannot compute boundary values) nor the Lagrangian formulation, used for structural computation (which cannot manage very large particles motion) are appropriate. To avoid these drawbacks, the ALE formulation has been introduced, first by [88] and [46]. In this formulation the referential coordinate called Ω_ξ, with system of coordinate $\underline{\xi}$, is used. This one is different from the spatial domain Ω_x. The mapping function between these two domains is called ϕ, such as:

$$\begin{aligned} \Phi : \Omega_\xi \times [0,T] &\rightarrow \Omega_x \times [0,T] \\ (\underline{\xi}, t) &\mapsto \Phi(\underline{\xi}, t) = (\underline{\mathbf{x}}, t) \end{aligned} \tag{1.7}$$

Then the ALE grid velocity $\underline{\mathbf{w}}$ is defined as:

$$\underline{\mathbf{w}} = \frac{\partial \underline{\mathbf{x}}}{\partial t}|_{\underline{\xi}} \tag{1.8}$$

The referential domain is also different from the material domain Ω_X. The mapping function between these two domains Ψ, is defined as:

$$\begin{aligned} \Psi : \Omega_\xi \times [0,T] &\rightarrow \Omega_X \times [0,T] \\ (\underline{\xi}, t) &\mapsto \Psi(\underline{\xi}, t) = (\underline{\mathbf{X}}, t) \end{aligned} \tag{1.9}$$

The mapping function φ can then be expressed as $\varphi = \Phi \circ \Psi^{-1}$, showing the relation between the three mappings φ, Φ and Ψ, see Fig.1.11.

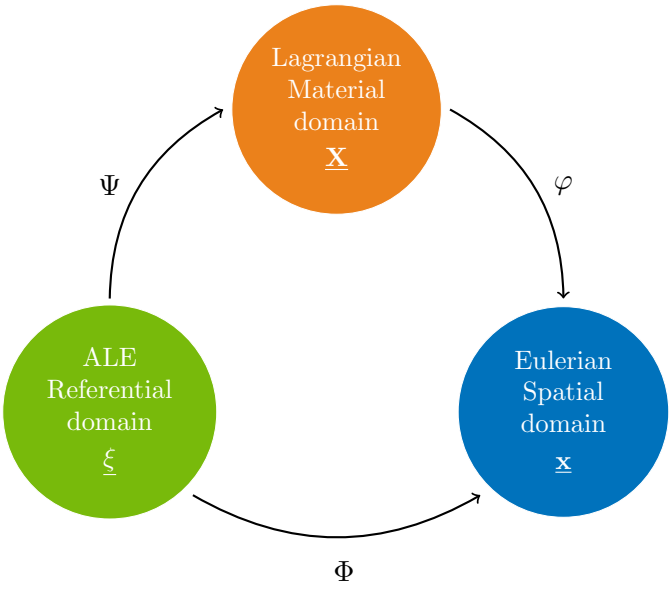


FIGURE 1.11: Referential, Material and Spatial domains transformations

To understand the advantages of the ALE formulation, let's move on to the discretized representation of this one, Fig.1.12. The idea of the ALE formulation is to use a moving grid, whose motions are arbitrary inside the domain, not connected with material points; except on the boundaries where the mesh node are confused with physical particles. In this way, the referential domain is allowed to fit moving walls. Moreover the arbitrary motions of the nodes inside the physical domain are chosen smoother than particles motion to avoid grid distortion.
Let's remark that, if the ALE grid velocity is chosen as $\underline{\mathbf{w}} = 0$, the formulation becomes eulerian. In the other way, using $\underline{\mathbf{w}} = \underline{\mathbf{v}}$, the Lagrangian formulation is found.

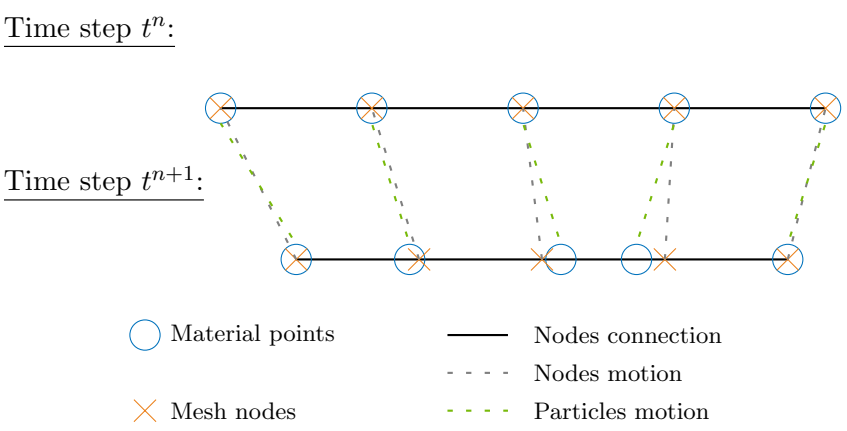


FIGURE 1.12: ALE grid motion between two time-steps

Thus, the ALE formulation is adapted for interface tracking fluid computation. Thanks to the Lagrangian behaviour of the ALE grid at boundaries, the motion of the interface Γ_{FS} can be computed. Moreover, thanks to the arbitrary motion of the ALE grid inside the domain, the grid distortion can be avoided. Yet, this formulation is

more costly than eulerian and lagrangian formulations because the grid motion has to be computed an updated at each time step. To reduce the computational cost, the arbitrary function of the ALE grid motion is often chosen really simple, as linear interpolation for example. Finally, if the interface motion is large, a re-meshing can still become necessary.

In this section, FSI reference problem was presented, as well as different approaches used to manage both fluid-structure formulations coupling and interface motion. This brief presentation of advantages and drawbacks of the main existing solutions, justifies the use of an interface tracking approach, using ALE formulation over fluid sub-domain and lagrangian formulation over the structural sub-domain, for the proposed coupling method.

1.3 Sub-domains decomposition state of the art

In this section coupling methods based sub-domains decomposition are described. First, we leave the field of fluid structure interaction, to propose a brief state of the art of solid sub-domains methods, which have been first developed. Then a review of the studies that have extended these methods to FSI problems is proposed.

1.3.1 Sub-domains decomposition methods for structural problems

The idea to divide a large and/or complex problem into several smaller problems is not recent and allows to gain computation time. They can be used for example to simulate two solid sub-domains with different characteristics as soil/structure simulation in [94]. They also can be used to divide the global problem in smaller sub-domains where different meshes are used according to their needs as in [4]. With these methods, an appropriate time step and a different time integrator can be used over each sub-domain [76, 50]. This is useful for example when a complex phenomenon occurs locally, and an explicit integrator and a smaller time step are required. Finally, these approaches are useful to parallelize the simulation and optimize computation time [120].
The decomposition in sub-domains can be with or without overlap. Here we are not talking about sub-domain decomposition methods with overlap, since we have already chosen an interface tracking approach for the proposed FSI methods. Then we considered a solid domain Ω divided without overlap into sub-domains. To be more readable, here, we consider a partition of only two sub-domains called Ω_1 and Ω_2, separated by the shared boundary Γ. The problem definition is the same as the one presented in section 1.2.1, replacing fluid and solid subscripts f and s by 1 and 2.

The two main approaches to solve this kind of problems are presented below; the primal approach [78, 79, 110, 70] and the dual approach [38, 40, 15].
The primal approaches are based on the kinematic continuity at the interface. The dynamic equilibrium is written on each sub-domain. Then, the kinematic variables, also called primal variables, here displacements, are used to impose the interface condition. Based on Finite Element Methods (FEM), described more in detail in the following chapter 2.1.1, the semi-discretized form of the problem can be written by

the system:

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{0} & \mathbf{A}_{1\Gamma} \\ \mathbf{0} & \mathbf{A}_{22} & \mathbf{A}_{2\Gamma} \\ \mathbf{A}_{\Gamma 1} & \mathbf{A}_{\Gamma 2} & \mathbf{A}_{\Gamma\Gamma} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{u}_\Gamma \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{f}_\Gamma \end{bmatrix} \tag{1.10}$$

Where $\mathbf{u}_1$ and $\mathbf{u}_2$ are internal nodal displacements inside the sub-domains Ω_1 and Ω_2 respectively. $\mathbf{u}_\Gamma$ are the nodal displacement of the interface. Then the system is solved by computing the interface displacements and then the internal displacements. The detail of resolution procedures can be found in [34].
The dual approaches are them based on kinematic and dynamic conditions. They impose synchronously kinematic and load equilibriums at the interface. The method is based on the stationarization of the variational formulation over the entire domain Ω, ensuring the kinematic continuity at the interface by the mean of Lagrange multipliers method. Using FEM, the semi-discretized, dual formulation of the problem can be written as:

$$\begin{bmatrix} \mathbf{A}_1 & \mathbf{0} & \mathbf{L}_1^T \\ \mathbf{0} & \mathbf{A}_2 & \mathbf{L}_2^T \\ \mathbf{L}_1 & \mathbf{L}_2 & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \\ \mathbf{\Lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \mathbf{0} \end{bmatrix} \tag{1.11}$$

Where $\mathbf{\Lambda}$ is the vector of the discretized interaction force. $\mathbf{L}_1$ and $\mathbf{L}_2$ are interface nodes selection vectors for nodes from sub-domain Ω_1 and from sub-domain Ω_2 respectively. The system is solved by computing the interface problem and then, the kinematic variables are solved. The procedure is detailed in [53] as example and is detailed later.
Let's compare primal and dual approaches. With primal approaches, the solution is sought in the function space where the interface condition is valid. While, with dual approaches, interface condition is induced in the weak formulation. Moreover, primal methods require to manipulate DOF inside the sub-domains to build system (1.10). Meanwhile, with a dual approach, that uses interface operators $\mathbf{L}_1$ and $\mathbf{L}_2$, the sub-domains are computed apart, except for the interface problem, which has a reasonable size as projected on the interface. This characteristic of dual approaches is a really good advantage in the scope of FSI simulation, when dedicated solver is to be used. Finally, dual methods enforce the load and the kinematic continuity at the interface synchronously, which is very well suited to enforce FSI equations (1.4a) and (1.4b). For these reasons, a domain decomposition dual approach is chosen for the proposed method.

A fast review of the coupling methods for dual approaches for solid domains decomposition is proposed below. Coupling methods for Schur's dual problem using FEM are called Finite Element Tearing and Interconnection (FETI) methods. Numerous FETI methods have been proposed and the first one has been introduced in [41], with proof of stability shown by [79]. Then, the method has been extended to solid dynamic [39]. A lot of improvements have been proposed, using different integration time, incompatibility in space and time [55, 28].
Concerning the simulation of sub-domains using different time scales, two approaches have been proposed. The first one, introduced by [53], called GC method, proposed to impose the coupling condition at the finer time scale. Moreover, authors have shown that the use of the continuity of velocity through the interface, as kinetic condition, leads to a stable algorithm, for mono-time scale and multi-time scales coupling. Thus, the mono-time scale GC method is energy preserving, while the multi-time step

method can induce energy dissipation at the interface, due to required interpolation at micro time step.
Later, the PH method has been introduced by [95, 96], where instead of imposing the coupling conditions at the micro time scale, they are enforced at the macro time scale. This method is more difficult to implement, more intrusive, but is energy preserving even with multi time scale. However, a lack of accuracy can be observed concerning the capturing of the interface motions, particularly if there are non-linear effects, due to the fact that coupling conditions are not computed at micro time step.
The objectives of the considered problem being interface tracking and the use of dedicated solvers, the proposed FSI coupling method is based on the GC method.

The end of the section is dedicated to succinctly describe the procedure of the GC method, for the coupling of two solid sub-domains, considered linear elastic, using the same time scale. The multi-scale method is used and presented in the next chapter 2.2.
The equilibrium equations in dual formulation, over each sub-domain, are spatially discretized using FEM and are temporally discretized using Newmark scheme [86].

$$\mathbf{M}_k\mathbf{a}_k^{n+1} + \mathbf{K}_k\mathbf{d}_k^{n+1} = \mathbf{F}_k^{n+1} + \mathbf{L}_k^T\mathbf{\Lambda}^{n+1} \tag{1.12a}$$

$$\mathbf{v}_k^{n+1} = \mathbf{v}_k^p + \gamma_k\Delta t\mathbf{a}_k^{n+1} \tag{1.12b}$$

$$\mathbf{d}_k^{n+1} = \mathbf{d}_k^p + \beta_k\Delta t^2\mathbf{v}_k^{n+1} \tag{1.12c}$$

Where $k = 1, 2$, the sub-domain subscript. $\mathbf{M}_k$ and $\mathbf{K}_k$ are mass and stiffness matrices respectively. The state vector at the current time step t^{n+1}, made on displacement, velocity and acceleration $[\mathbf{d}_k^{n+1}, \mathbf{v}_k^{n+1}, \mathbf{a}_k^{n+1}]$ is searched from the previous step known at t^n. $t^{n+1} = \Delta t + t^n$, where Δt is the common time step used for both sub-domains. $\mathbf{d}_k^p$ and $\mathbf{v}_k^p$ are Newmark predictors see equations (2.32) and (2.33). Finally γ_k and β_k are Newmark parameters. For instance, the temporal integration scheme is explicit for $\gamma_k = \frac{1}{2}$ and $\beta_k = 0$ and implicit for $\gamma_k = \frac{1}{2}$ and $\beta_k = \frac{1}{4}$.
Then the coupling condition can also be written in a discretized form:

$$\mathbf{L}_1\mathbf{v}_1^{n+1} + \mathbf{L}_2\mathbf{v}_2^{n+1} = \mathbf{0} \tag{1.13}$$

Let's define the Steklov Poincaré operator as:

$$\mathbf{H}_{12} = \gamma_1\Delta t\mathbf{L}_1\tilde{\mathbf{M}}_1^{-1}\mathbf{L}_1^T + \gamma_2\Delta t\mathbf{L}_2\tilde{\mathbf{M}}_2^{-1}\mathbf{L}_2^T \tag{1.14}$$

Where $\tilde{\mathbf{M}}_k = \mathbf{M}_k + \beta_k\Delta t^2\mathbf{K}_k$.
Then, including equation (1.12b) into equation (1.13) and using operator (1.14), Lagrange multipliers are expressed as:

$$\mathbf{H}_{12}\mathbf{\Lambda}^{n+1} = \mathbf{L}_1\mathbf{v}_{1_{free}}^{n+1} + \mathbf{L}_2\mathbf{v}_{2_{free}}^{n+1} \tag{1.15}$$

$\mathbf{v}_{k_{free}}^{n+1}$ are introduced as the *free* nodal velocities of the sub-domains Ω_k. *Free* variables come from the computation of each sub-domain alone, without taking any interaction between them into account, only materials history and boundary conditions are used. Yet, the state vectors can be split into a *free* part and a *link* part. The *link* state

vectors being the part of state vector which depends on interface interactions.

$$\begin{bmatrix} \mathbf{a}_k^n \\ \mathbf{v}_k^n \\ \mathbf{d}_k^n \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{k_{free}}^n \\ \mathbf{v}_{k_{free}}^n \\ \mathbf{d}_{k_{free}}^n \end{bmatrix} + \begin{bmatrix} \mathbf{a}_{k_{link}}^n \\ \mathbf{v}_{k_{link}}^n \\ \mathbf{d}_{k_{link}}^n \end{bmatrix} \tag{1.16}$$

Thus, the system (1.12) is split into the *free* problem and the *link* problem for both sub-domains. The *free* state vectors are computed such as:

$$\mathbf{a}_{k_{free}}^{n+1} = \tilde{\mathbf{M}}_k^{-1}(\mathbf{F}_k^{n+1} - \mathbf{K}_k \mathbf{d}_k^p) \tag{1.17a}$$

$$\mathbf{v}_{k_{free}}^{n+1} = \mathbf{v}_k^p + \gamma_k \Delta t \mathbf{a}_{k_{free}}^{n+1} \tag{1.17b}$$

$$\mathbf{d}_{k_{free}}^{n+1} = \mathbf{d}_k^p + \beta_k \Delta t^2 \mathbf{a}_{k_{free}}^{n+1} \tag{1.17c}$$

And *link* state vectors are defined as:

$$\mathbf{a}_{k_{link}}^{n+1} = \tilde{\mathbf{M}}_k^{-1}(-\mathbf{L}_k^T \mathbf{\Lambda}^{n+1}) \tag{1.18a}$$

$$\mathbf{v}_{k_{link}}^{n+1} = \gamma_k \Delta t \mathbf{a}_{k_{link}}^{n+1} \tag{1.18b}$$

$$\mathbf{d}_{k_{link}}^{n+1} = \beta_k \Delta t^2 \mathbf{a}_{k_{link}}^{n+1} \tag{1.18c}$$

Finally, the GC method solves this solid problem which is decomposed into two sub-domains discretized by FEM and Newmark scheme implicit or explicit, with the same time step, following the procedure described by Alg.1.

Algorithm 1 GC method procedure

1: Initialization of Ω_1 and Ω_2
2: Compute invariants : $\tilde{\mathbf{M}}_1, \tilde{\mathbf{M}}_2, \mathbf{H}_{12} \leftarrow (1.14)$
3: **while** $t^n \leq T$ **do** ▷ Time loop
4: Compute $\mathbf{U}_{k_{free}}^{n+1} \leftarrow (1.17)$
5: Compute $\mathbf{\Lambda}^{n+1} \leftarrow (1.15)$
6: Compute $\mathbf{U}_{k_{link}}^{n+1} \leftarrow (1.18)$
7: Compute $\mathbf{U}_k^{n+1} \leftarrow (1.16)$
8: **end while**

This section presented a brief review of the methods developed for structural domains decomposition, with their advantages and drawbacks. The idea of the proposed coupling method is to extend this kind of method to FSI problems. Regarding the FSI issues considered; conservative coupling and use of dedicated solvers, the dual approach and the GC methods are the more appropriate.

1.3.2 Sub-domains decomposition for FSI problems

This section presents a review of the studies which have proposed to extend the dual approach for structural domains decomposition methods to FSI problem.
The first method has been proposed by Casadei [21] in 2011. This work is based on FETI method with mono-time scale. The solid sub-domain is discretized in space by FEM, in Lagrangian formulation. The temporal discretization is led by an explicit

central difference scheme. The fluid sub-domain is discretized by vertex-centered Finite Volumes method in ALE formulation. Concerning temporal discretization, an explicit backward Euler scheme is used with MUSCL-like technique in order to reach order two in time. Finally, the coupling conditions are imposed by the mean of Lagrange multipliers which impose every time step the continuity of velocities at the interface. Then, the author proposes a unified scheme of resolution for FE and FV. Concerning the ALE grid variables, those are solved at mid-step using an explicit formulation of their velocities. At full-time step, the ALE grid velocities are unknown. That is why they are approximated as their values from the mid-step, and the ALE grid is updated from here. The coupling method is based on a dual monolithic formulation, couples FE and FV vertex centered, with explicit schemes and a mono-time scale.

Li proposes in [73] a coupling method for FSI also based on dual approach of domain decomposition, with mono time scale. The solid sub-domain is discretized using Finite Elements and Newmark implicit scheme. In order to ease large displacements of the interface, the discretization method chosen is not the common FVM for fluid problem, but SPH which is a mesh-less method. Nevertheless, the ALE formulation is still used. For the temporal discretization, an explicit Runge-Kutta order 2 scheme is used. The same temporal scale is used over both sub-domains. The velocities continuity is enforced at mid-step and full time step, using Lagrange multipliers. The method is energy preserving. Contrary to the method proposed by Casadei, here, there is no need to approximate the ALE grid quantities. Indeed, thanks to the SPH discretization, only the grid velocity appears in fluid computation and the numerical flux being explicit; the ALE velocities are always known. Thus, the method is based on a dual monolithic formulation. It is heterogeneous coupling explicit SPF fluid and implicit FE solid, and mono-time scale.

Later, Nunez Ramirez [89], proposes an extension on the work of Li to multi-time scales resolution, based on the GC method. The solid sub-domain is discretized by FEM and explicit Newmark scheme. The fluid sub-domain is discretized by ALE-SPH and explicit Runge-Kutta scheme. Both sub-domains are discretized according to their proper time step. Then, the continuity of the velocities through the interface is imposed at the finer time-scale, as in the GC method. To sum up, in this work, the method couples solid FE and fluid SPH-ALE, fully explicit, with multi-time scale.

Finally, Meduri [81], proposes also an FSI coupling based on multi-time scale GC method but uses different formulation and discretization methods. In this work, the solid and the fluid sub-domains use a Lagrangian formulation. The solid sub-domain is discretized by FEM and the fluid sub-domain is discretized by Particle Finite Element Method (PFEM), which is more adapted to fluid flow. The temporal discretization is led over each sub-domain by the explicit central difference scheme, but with different time step sizes. Finally, the velocity continuity is imposed at the finer time-scale. In this way, the coupling method is fully explicit and Lagrangian, coupling FE and PFE, with multi time scale.

Conclusion

In this first chapter, generalities about FSI problems have been presented, in order to explain our motivations and justify the chosen strategy of the proposed method in this work.
First, a classification of coupling methods according to physical coupling and numerical coupling based on the formulation was proposed. As mentioned in the introduction, the main objective of this work is to propose a preserving and stable coupling method. Therefore, a monolithic formulation is used. Though, the advantages induced by the implementation of the algorithms used for partitioned formulation are very attractive. Then, the idea of the proposed method is to solve the monolithic formulation, in order to ensure the global stability of the coupling, with a co-simulation algorithm, in the style of the partitioned approaches resolution.
Then the issues induced by the fluid-structure interface have been highlighted. The more popular approaches have been presented. Knowing their advantages and drawbacks, an interface tracking approach managed by an ALE formulation on the fluid sub-domain is chosen.
Finally, the main domain decomposition methods for solid problems have been presented as well as a review of the extension of these approaches to FSI simulation. The proposed method is based on the extension of the GC method, as in some previous works. The dual approach allows to account synchronously for pressure and velocities at the interface. Moreover, the enforcing of coupling conditions at the macro time-step is easier to implement which is more adapted to co-simulation.
The objective is then to propose a version of the GC method for FSI coupling with discretization methods used in existing solvers, in order to propose a code coupling solution. In this way, the fluid sub-domain uses FVM cell centered and the solid sub-domain uses FEM. Moreover, we want to propose a heterogeneous coupling method, using an implicit temporal scheme on solid and an explicit scheme on fluid. This coupling method is presented in the next chapter.

Chapter 2

Towards a stable non-intrusive FSI coupling method

The aim of this chapter is to theoretically present the proposed coupling method. As exposed in the section 1.3 the idea is to extend the GC method to fluid-structure interaction problems with classical discretization methods on each sub-domain in order to use dedicated solvers with their own time scale. The solid sub-domain is discretized using the Finite Element Method and an implicit Newmark scheme, as it is done in Ansys Mechanical [6] or CalculiX [33] for example. Regarding the fluid sub-domain, the spatial discretization is done thanks to the cell centered finite volume method, and a temporal second order Runge-Kutta scheme is used. These discretization methods are both available in Ansys Fluent [7]. First, the equations solved and the associated chosen spatial and temporal discretization methods are described on each sub-domain separately. Then, the proposed coupling method is presented from the monolithic formulation to the algorithm of resolution. Finally, the results of the method are studied on the academic test case of the 1D piston.

2.1 Studied physical system and local equations

The first step is to describe the equations solved on each sub-domain. Let's recall that the method chosen for the mobile fluid-structure interface treatment is an interface capturing method, see section 1.2.1. Thus, here a domain Ω closed by a boundary Γ is considered. This domain is divided into two sub-domains Ω_s and Ω_f without overlap, such as $\Omega_s \cup \Omega_f = \Omega$ and $\Omega_s \cap \Omega_f = \emptyset$. The boundary between the fluid sub-domain Ω_f and the solid sub-domain Ω_s is named Γ_{FS}, that is the Fluid-Structure interface, see Fig.(1.7). The behavior of the domain Ω is studied from the initial time $t^0 = 0$ to the final time of the simulation $t^f = T$.

2.1.1 Solid sub-domain

First, we consider, as shown in Fig.2.1, the solid sub-domain Ω_s alone. It is subjected to a volumic force $\underline{\mathbf{f}}_s$ in Ω_s and a contact force $\underline{\mathbf{f}}_{ext}$ on Γ_N. The displacement $\underline{\mathbf{d}}_D$ is imposed on Γ_D. Γ_N and Γ_N are the Neumann and Dirichlet boundary conditions respectively, such as $\Gamma_N \cup \Gamma_D = \Gamma_s$ and $\Gamma_N \cap \Gamma_D = \emptyset$.

Continuous problem

The solid sub-domain is homogeneous, continuous and isotropic. The studied material is considered as linear elastic under small deformation. The theory of elasticity [49]

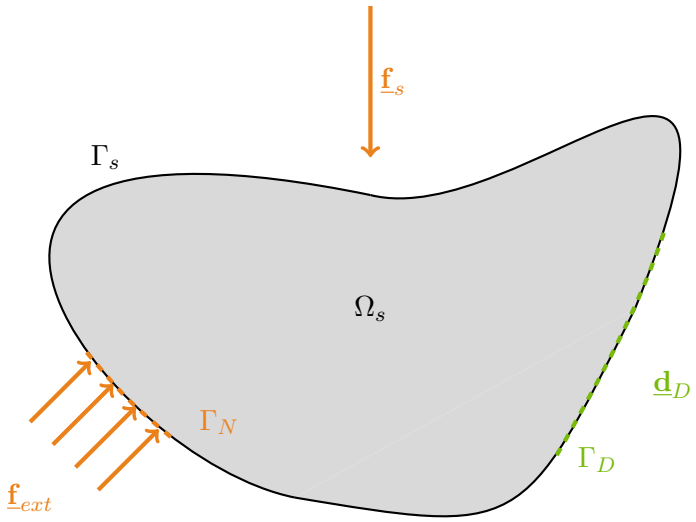


FIGURE 2.1: Solid sub-domain definition

allows us to write the strong formulation of the problem. The following continuous equations system is written in $\Omega_s \times [0,T]$, in the Lagrangian frame, to determine the field of displacement $\underline{\mathbf{d}}_s$.

$$\begin{cases} \underline{\nabla} \cdot \underline{\underline{\sigma}}_s(\underline{\mathbf{X}},t) + \underline{\mathbf{f}}_s(\underline{\mathbf{X}},t) = \rho_s \dfrac{\partial^2 \underline{\mathbf{d}}_s(\underline{\mathbf{X}},t)}{\partial t^2} & (\underline{\mathbf{X}},t) \in \Omega_s \times [0,T] \quad (2.1a) \\ \underline{\underline{\sigma}}_s(\underline{\mathbf{X}},t) = \underline{\underline{\underline{\underline{\mathbf{K}}}}} : \underline{\underline{\epsilon}}(\underline{\mathbf{X}},t) & (\underline{\mathbf{X}},t) \in \Omega_s \times [0,T] \quad (2.1b) \\ \underline{\mathbf{d}}_s(\underline{\mathbf{X}},t) = \underline{\mathbf{d}}_D(\underline{\mathbf{X}},t) & (\underline{\mathbf{X}},t) \in \Gamma_D \times [0,T] \quad (2.1c) \\ \underline{\underline{\sigma}}_s(\underline{\mathbf{X}},t) \cdot \underline{\mathbf{n}}(\underline{\mathbf{X}}) = \underline{\mathbf{f}}_{ext}(\underline{\mathbf{X}},t) & (\underline{\mathbf{X}},t) \in \Gamma_N \times [0,T] \quad (2.1d) \\ \underline{\mathbf{d}}_s(\underline{\mathbf{X}},0) = \underline{\mathbf{d}}_s^0(\underline{\mathbf{X}}), \quad \underline{\mathbf{v}}_s(\underline{\mathbf{X}},0) = \underline{\mathbf{v}}_s^0(\underline{\mathbf{X}}) & \underline{\mathbf{X}} \in \Omega_s \quad (2.1e) \end{cases}$$

In order to be more readable, the space-time dependency notations, $(\underline{\mathbf{X}},t)$, are dropped. The first two equations are the linear momentum equation and the constitutive law, where $\underline{\underline{\sigma}}_s$ is the symmetric stress tensor, $\underline{\underline{\underline{\underline{\mathbf{K}}}}}$ is the Hooke tensor and $\underline{\underline{\epsilon}}$ is the linearized strain tensor defined as:

$$\underline{\underline{\epsilon}} = \frac{1}{2}\left((\underline{\nabla}\,\underline{\mathbf{d}}_s) + (\underline{\nabla}\,\underline{\mathbf{d}}_s)^T\right) \tag{2.2}$$

The next two equations are the Dirichlet and Neumann boundary conditions, respectively. The Dirichlet boundary conditions, also called essential conditions, impose the kinematic variables, while the Neumann boundary conditions, also called natural conditions, prescribe load quantities. The last two equations are the initial conditions, with $\underline{\mathbf{d}}_s^0$ and $\underline{\mathbf{v}}_s^0$ the initial displacement and velocity, respectively.

The Finite Element Method, used for the spatial discretization, is based on the weak

formulation whose existence and uniqueness of its solution are studied in [35]. Moreover, in [57] the equivalence between the strong formulation (2.1) and the weak formulation is demonstrated. The weak formulation can be obtained by the stationarization of the Lagrangian, defined as (2.3), also called Hamilton principle [1].

$$\mathscr{L}_s(\underline{\mathbf{d}}_s) = \mathscr{K}(\underline{\dot{\mathbf{d}}}_s) - \left(\mathscr{W}_{int}(\underline{\mathbf{d}}_s) - \mathscr{W}_{ext}(\underline{\mathbf{d}}_s)\right) \tag{2.3}$$

Such as:

$$\delta \int_0^T \mathscr{L}_s(\underline{\mathbf{d}}_s)\,\mathrm{d}t = 0 \tag{2.4}$$

Where $\mathscr{K}$, $\mathscr{W}_{int}$ and $\mathscr{W}_{ext}$ are the kinetic energy, the internal energy and the external energy, respectively, defined in linear elasticity such as:

$$\mathscr{K}(\underline{\dot{\mathbf{d}}}_s) = \int_{\Omega_s} \frac{1}{2}\rho_s \underline{\dot{\mathbf{d}}}_s \cdot \underline{\dot{\mathbf{d}}}_s \,\mathrm{d}\Omega_s \tag{2.5}$$

$$\mathscr{W}_{int}(\underline{\mathbf{d}}_s) = \int_{\Omega_s} \frac{1}{2}\underline{\underline{\underline{\underline{\mathbf{K}}}}}.\underline{\underline{\epsilon}}(\underline{\mathbf{d}}_s) : \underline{\underline{\epsilon}}(\underline{\mathbf{d}}_s)\,\mathrm{d}\Omega_s \tag{2.6}$$

$$\mathscr{W}_{ext}(\underline{\mathbf{d}}_s) = \int_{\Omega_s} \underline{\mathbf{f}}_s \cdot \underline{\mathbf{d}}_s \,\mathrm{d}\Omega_s + \int_{\Gamma_N} \underline{\mathbf{f}}_{ext} \cdot \underline{\mathbf{d}}_s \,\mathrm{d}\Gamma_N \tag{2.7}$$

The weak formulation is also linked to the principle of virtual work [48]. This one expresses that the sum of the internal virtual work called P_i^* and the virtual work of the external forces called P_e^* are equal to the virtual work of the acceleration quantities called P_a^*.
The internal virtual work is defined as:

$$P_i^* = -\int_{\Omega_s} \underline{\underline{\sigma}}_s(\underline{\mathbf{d}}_s) : \underline{\underline{\epsilon}}(\delta\underline{\mathbf{X}}_s)\,\mathrm{d}\Omega_s \tag{2.8}$$

The virtual work of the external forces is the sum of the work of the volumic force $\underline{\mathbf{f}}_s$ applied in the solid sub-domain Ω_s and the surface forces $\underline{\mathbf{f}}_{ext}$ applied on Γ_N:

$$P_e^* = \int_{\Omega_s} \underline{\mathbf{f}}_s \cdot \delta\underline{\mathbf{X}}_s \,\mathrm{d}\Omega_s + \int_{\Gamma_N} \underline{\mathbf{f}}_{ext} \cdot \delta\underline{\mathbf{X}}_s \,\mathrm{d}\Gamma_N \tag{2.9}$$

Finally, the virtual work of the acceleration quantities is defined as:

$$P_a^* = \int_{\Omega_s} \rho_s \frac{\partial^2 \underline{\mathbf{d}}_s}{\partial t^2}.\delta\underline{\mathbf{X}}_s \,\mathrm{d}\Omega_s \tag{2.10}$$

Let us define $\mathscr{U}$, the set of the admissible solutions; the field of displacement $\underline{\mathbf{d}}_s$, which respects the Dirichlet boundary conditions, is sufficiently continuous and regular, in the functional space $\mathscr{H}^1$:

$$\int_{\Omega_s} (\underline{\mathbf{d}}_s)^2 \, d\Omega_s < +\infty \qquad \int_{\Omega_s} (\frac{\partial \underline{\mathbf{d}}_s}{\partial x})^2 \, d\Omega_s < +\infty \tag{2.11}$$

We define $\mathscr{U} = \{\underline{\mathbf{d}}_s \,|\, \underline{\mathbf{d}}_s \in \mathscr{H}^1, \underline{\mathbf{d}}_s = \underline{\mathbf{d}}_D \text{ on } \Gamma_D\}$. In the same way, we define $\mathscr{V}^0$ the set of the test functions kinematically admissible at 0 such as $\mathscr{V}^0 = \{\delta\underline{\mathbf{X}}_s \,|\, \delta\underline{\mathbf{X}}_s \in$

$\mathscr{H}^1$, $\delta\underline{\mathbf{X}}_s = \underline{\mathbf{0}}$ on $\Gamma_D\}$. Finally, for all $\delta\underline{\mathbf{X}}_s \in \mathscr{V}^0$, the principle of virtual work expresses as:

$$P_i^* + P_e^* - P_a^* = 0 \tag{2.12}$$

To move to the weak formulation in space and in time of the problem, the principle of virtual work (2.12) is integrated in time. The displacement field $\underline{\mathbf{d}}_s \in \mathscr{U}$ is searched for all $\delta\underline{\mathbf{X}}_s \in \mathscr{V}^0$, such as:

$$\int_0^T P_i^* + P_e^* - P_a^* \, \mathrm{d}t = 0 \tag{2.13}$$

Where $\mathscr{U} = \{\underline{\mathbf{d}}_s \,|\, \underline{\mathbf{d}}_s \in \mathscr{H}^1, \underline{\mathbf{d}}_s(\underline{\mathbf{X}}, t) = \underline{\mathbf{d}}_D(\underline{\mathbf{X}}, t) \text{ on } \Gamma_D \times [0,T], \underline{\mathbf{d}}_s(\underline{\mathbf{X}}, 0) = \underline{\mathbf{d}}_s^0(\underline{\mathbf{X}}) \text{ in } \Omega_s, \underline{\mathbf{v}}_s(\underline{\mathbf{X}}, 0) = \underline{\mathbf{v}}_s^0(X) \text{ in } \Omega_s\}$ and $\mathscr{V}^0 = \{\delta\underline{\mathbf{X}}_s \,|\, \delta\underline{\mathbf{X}}_s \in \mathscr{H}^1, \delta\underline{\mathbf{X}}_s(\underline{\mathbf{X}}, t) = 0 \text{ on } \Gamma_D \times [0,T], \delta\underline{\mathbf{X}}_s(X, 0) = 0 \text{ in } \Omega_s, \delta\underline{\mathbf{X}}_s(\underline{\mathbf{X}}, T) = 0 \text{ on } \Omega_s\}$.

Spatial dicretization

To solve this kind of continuous problem, the idea is to solve an approximated problem on a discretized space. Here, we use the Finite Element Method, as traditionally for solid problems [57]. The continuous space is approximated by a finite number n_{el} of geometrical elements Ω_s^E, which are most of the time triangles or quadrangles in 2D for example and whose vertices are called nodes, see Fig. 2.2.

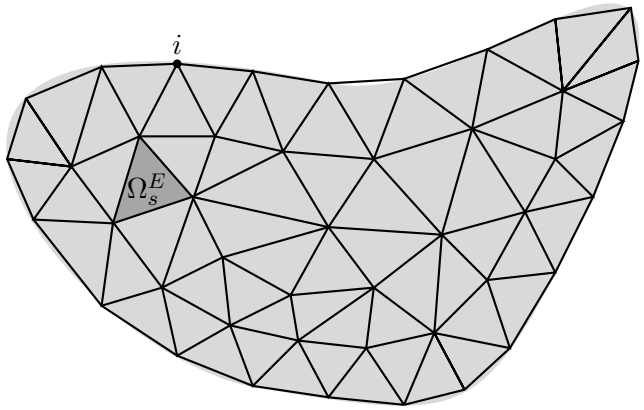


FIGURE 2.2: Solid sub-domain spatial discretization

Then, the continuous fields are discretized from the nodal quantities; the admissible solution $\mathscr{U}$ is approximated by $\mathscr{U}_n$ and the virtual functional $\mathscr{V}^0$ is approximated by $\mathscr{V}_n^0$. Then, the equation (2.13) of the weak formulation of the equilibrium can be approximated by the discretized Galerkin formulation. The displacement field is obtained by the resolution at the nodes of the equilibrium equation. Then the nodal displacements are interpolated over each finite element according to their shape

functions, and we define:

$$\underline{\mathbf{d}}_s(\underline{\mathbf{X}},t) \approx \sum_{i=1}^{n_s} \underline{\mathbf{N}}_i(\underline{\mathbf{X}})\mathbf{d}_{s_i}(t) \tag{2.14}$$

Where $i \in [1,...,n_s]$ the index of the nodes of the mesh considered in its global classification. $\underline{\mathbf{N}}_i$ is the shape function associated to the node i.
The same interpolation is applied to the velocity field. By derivation, we get:

$$\underline{\mathbf{v}}_s(\underline{\mathbf{X}},t) \approx \frac{\partial(\sum_{i=1}^{n_s} \underline{\mathbf{N}}_i(\underline{\mathbf{X}})\mathbf{d}_{s_i}(t))}{\partial t} = \sum_{i=1}^{n_s} \underline{\mathbf{N}}_i(\underline{\mathbf{X}})\mathbf{v}_{s_i}(t) \tag{2.15}$$

Finally, using Galerkin approach, the same discretization is assumed for the virtual field, with $\delta\mathbf{X}_{s_i}$ denoting the virtual degrees of freedoms associated with the node i.

$$\delta\underline{\mathbf{X}}_s(\underline{\mathbf{X}},t) \approx \sum_{i=1}^{n_s} \underline{\mathbf{N}}_i(\underline{\mathbf{X}})\delta\mathbf{X}_{s_i}(t) \tag{2.16}$$

Thereby, using (2.14), (2.15) and (2.1b) into (2.13) the discretized weak formulation of equilibrium, using Einstein convention, is written as:

$$\int_0^T \left[\delta\mathbf{X}_{s_i}^T \int_{\Omega_s} \rho_s \underline{\mathbf{N}}_i^T \cdot \underline{\mathbf{N}}_j \,\mathrm{d}\Omega_s \mathbf{a}_{s_j} + \delta\mathbf{X}_{s_i}^T \int_{\Omega_s} \underline{\underline{\underline{\mathbf{K}}}} : \underline{\underline{\epsilon}}(\underline{\mathbf{N}}_i) : \underline{\underline{\epsilon}}(\underline{\mathbf{N}}_j) \,\mathrm{d}\Omega_s \mathbf{d}_{s_j}\right] \mathrm{d}t = \int_0^T \left[\delta\mathbf{X}_{s_i}^T \int_{\Omega_s} \underline{\mathbf{N}}_i \cdot \underline{\mathbf{f}}_{s_i} \,\mathrm{d}\Omega_s + \int_{\Gamma_N} \underline{\mathbf{N}}_i \cdot \underline{\mathbf{f}}_{ext_i} \,\mathrm{d}\Gamma_N\right] \mathrm{d}t \tag{2.17}$$

Then the mass matrix $\mathbf{M}_s$, of size $(n_s \times n_s)$ symmetric and positive definite is written as:

$$M_{s_{ij}} = \sum_{E=1}^{n_{el}} \int_{\Omega_s^E} \rho_s \underline{\mathbf{N}}_i^E \cdot \underline{\mathbf{N}}_j^E \,\mathrm{d}\Omega_s^E \qquad i,j \in [1,...,n_s] \tag{2.18}$$

Moreover, thanks to the hypothesis of linear elasticity of the material, the expression of the internal forces can be written as the following vector:

$$\mathbf{F}_{\mathbf{int}_i} = \int_{\Omega_s} \underline{\underline{\underline{\mathbf{K}}}} : \underline{\underline{\epsilon}}(\underline{\mathbf{N}}_i) : \underline{\underline{\epsilon}}(\underline{\mathbf{N}}_j) \,\mathrm{d}\Omega_s \mathbf{d}_{s_j} = \mathbf{K}_{s_{ij}}\mathbf{d}_{s_j} \tag{2.19}$$

Where $\underline{\mathbf{N}}_i$ and $\underline{\mathbf{N}}_j$ are the matrices of the shape functions related to the node i and j, respectively. $\mathbf{K}_s$ is the symmetric stiffness matrix defined such as:

$$K_{s_{ij}} = \sum_{E=1}^{n_{el}} \int_{\Omega_s^E} \underline{\underline{\underline{\mathbf{K}}}} : \underline{\underline{\epsilon}}(\underline{\mathbf{N}}_i^E) : \underline{\underline{\epsilon}}(\underline{\mathbf{N}}_j^E) \,\mathrm{d}\Omega_s^E \qquad i,j \in [1,...,n_s] \tag{2.20}$$

Finally, the equation (2.17) being true for all $\delta\mathbf{X}_s \in \mathscr{V}_n^0$, the matrix form of the semi-discretized linear momentum equation is written as:

$$\mathbf{M}_s\mathbf{a}_s(t) + \mathbf{K}_s\mathbf{d}_s(t) = \mathbf{F}_s(t) \qquad \forall t \in [0,T] \tag{2.21}$$

Time discretization

The equation (2.21) is now discretized in space, but continuous in time. The resolution of this kind of structural dynamic problem is made by implicit or explicit time integration schemes. First the continuous time interval $[0, T]$ is divided into a chosen and finite number of time steps, called Δt_s, see Fig.2.3. Then the integration scheme is used to compute the solution of the problem only at the discrete times.

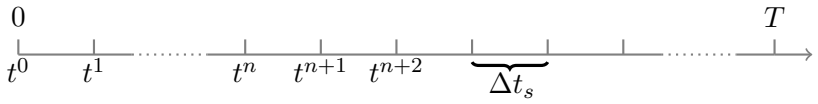


FIGURE 2.3: Time discretization

In this work, the time scheme chosen is the well-known Newmark scheme [86]. These kinds of scheme are one step schemes, where the solution at t^{n+1}, $t^{n+1} = t^n + \Delta t_s$, is computed thanks to the state at the previous time t^n and boundary conditions at the current time step t^{n+1}.
First, the displacements at the current time step are expressed from the displacements at the previous time and the integral term of the velocity across the time step:

$$\mathbf{d}_s(t^n + \Delta t_s) = \mathbf{d}_s(t^n) + \int_{t^n}^{t^{n+1}} \mathbf{v}_s(\tau)\,\mathrm{d}\tau \tag{2.22}$$

Then, adding a factor 1 under the integral and applying an integration by parts, considering the integral of the factor 1 as $(\tau - t^{n+1})$, the equation (2.22) becomes:

$$\mathbf{d}_s(t^n + \Delta t_s) = \mathbf{d}_s(t^n) - \Big[(t^{n+1} - \tau)\mathbf{v}_s(\tau)\Big]_{t^n}^{t^{n+1}} + \int_{t^n}^{t^{n+1}} (t^{n+1} - \tau)\mathbf{a}_s(\tau)\,\mathrm{d}\tau \tag{2.23}$$

Recalling the notation of the time step $\Delta t_s = t^{n+1} - t^n$ and considering the following notation for the discretized fields in space and time $\mathbf{d}_s(t^n) = \mathbf{d}_s^n$, the equation (2.23) is re-written as:

$$\mathbf{d}_s^{n+1} = \mathbf{d}_s^n + \Delta t_s \mathbf{v}_s^n + \int_{t^n}^{t^{n+1}} (t^{n+1} - \tau)\mathbf{a}_s(\tau)\,\mathrm{d}\tau \tag{2.24}$$

The velocities are now written in the same way as the displacements in (2.22) :

$$\mathbf{v}_s^{n+1} = \mathbf{v}_s^n + \int_{t^n}^{t^{n+1}} \mathbf{a}_s(\tau)\,\mathrm{d}\tau \tag{2.25}$$

Now the integrals of the accelerations have to be approximated. Newmark approximation are given using the following weighting from the values at the beginning and the end of the time step:

$$\int_{t^n}^{t^{n+1}} (t^{n+1} - \tau)\mathbf{a}_s(\tau)\,\mathrm{d}\tau \;\approx\; (\frac{1}{2} - \beta)\Delta t_s^2 \mathbf{a}_s^n + \beta \Delta t_s^2 \mathbf{a}_s^{n+1} \tag{2.26}$$

$$\int_{t^n}^{t^{n+1}} \mathbf{a}_s(\tau)\,\mathrm{d}\tau \;\approx\; (1-\gamma)\Delta t_s \mathbf{a}_s^n + \gamma \Delta t_s \mathbf{a}_s^{n+1} \tag{2.27}$$

Where $0 < \gamma < 1$ and $0 < \beta < \frac{1}{2}$. With the $\gamma = 0$ and $\beta = 0$, the weighting is fully backward while with $\gamma = 1$ and $\beta = \frac{1}{2}$ it is fully forward.
Then, including (2.26) and (2.27) into the equations (2.24) and (2.25) respectively, we get the discretized equations of displacements and velocities, corresponding to the well known Newmark approximation:

$$\mathbf{d}_s^{n+1} = \mathbf{d}_s^n + \Delta t_s \mathbf{v}_s^n + (\frac{1}{2} - \beta)\Delta t_s^2 \mathbf{a}_s^n + \beta \Delta t_s^2 \mathbf{a}_s^{n+1} \tag{2.28}$$

$$\mathbf{v}_s^{n+1} = \mathbf{v}_s^n + (1-\gamma)\Delta t_s \mathbf{a}_s^n + \gamma \Delta t_s \mathbf{a}_s^{n+1} \tag{2.29}$$

Finally, the last part of the Newmark scheme is the equilibrium equation. The equation (2.21) is re-written at the time t^{n+1}:

$$\mathbf{M}_s \mathbf{a}_s^{n+1} + \mathbf{K}_s \mathbf{d}_s^{n+1} = \mathbf{F}_s^{n+1} \tag{2.30}$$

Replacing $\mathbf{d}_s^{n+1}$ by its value from (2.28) and reorganizing the equation (2.30), we get:

$$\big(\mathbf{M}_s + \beta \Delta t_s^2 \mathbf{K}_s\big)\mathbf{a}_s^{n+1} = \mathbf{F}_s^{n+1} - \mathbf{K}_s\big(\mathbf{d}_s^n + \Delta t_s \mathbf{v}_s^n + (\frac{1}{2} - \beta)\Delta t_s^2 \mathbf{a}_s^n\big) \tag{2.31}$$

Let us introduce the following notations:

$$\mathbf{d}_s^p = \mathbf{d}_s^n + \Delta t_s \mathbf{v}_s^n + (\frac{1}{2} - \beta)\Delta t_s^2 \mathbf{a}_s^n \tag{2.32}$$

$$\mathbf{v}_s^p = \mathbf{v}_s^n + (1-\gamma)\Delta t_s \mathbf{a}_s^n \tag{2.33}$$

$$\tilde{\mathbf{M}}_s = \mathbf{M}_s + \beta \Delta t_s^2 \mathbf{K}_s \tag{2.34}$$

$\mathbf{d}_s^p$ and $\mathbf{v}_s^p$ are called displacement and velocity predictors, respectively.

The parameters γ and β are chosen according to convergence, stability, consistently and conservation criteria. They also determine the kind of integration scheme used. There are two main kinds of Newmark schemes:

- Explicit scheme : $\gamma = \frac{1}{2}$ and $\beta = 0$

This scheme is of order two, symplectic and energy conservative. It is the most interesting in terms of efficiency. Indeed, when $\beta = 0$ the displacements at t^{n+1} are known directly from the state at t^n. Moreover, $\tilde{\mathbf{M}}_s = \mathbf{M}_s$ and the mass matrix is lumped to be diagonal. In this way the resolution of the accelerations is straight forward. Nevertheless, this scheme is conditionally stable, the used time step Δt_s should not be superior to the critical time step $\Delta t_c = \frac{h}{c}$, with the h geometrical element size and c the P-wave, where $c^2 = \frac{E}{\rho_s}$ for the 1D case. This scheme is well suited for fast transient dynamics as crash for instance.

- Implicit scheme : $\gamma = \frac{1}{2}$ and $\beta = \frac{1}{4}$

This scheme is also order two and conservative, and unlike the explicit scheme, it is unconditionally stable; as a consequence, any Δt_s can be chosen according to the load evolution. It is well adapted for slow dynamics as earthquake for example.

An implicit time integrator scheme is adopted, since the considered solid material is linear elastic.
Thus, the computation uses the following procedure. Considering that the state at t^n is already known, the goal is to calculate displacements $\mathbf{d}_s^{n+1}$, velocities $\mathbf{v}_s^{n+1}$ and accelerations $\mathbf{a}_s^{n+1}$ at the following time t^{n+1}.
In a first stage, displacements and velocities predictors are computed as:

$$\mathbf{d}_s^p = \mathbf{d}_s^n + \Delta t_s \mathbf{v}_s^n + \frac{1}{4}\Delta t_s^2 \mathbf{a}_s^n \tag{2.35a}$$

$$\mathbf{v}_s^p = \mathbf{v}_s^n + \frac{1}{2}\Delta t_s \mathbf{a}_s^n \tag{2.35b}$$

In a second stage, the equilibrium equation is solved to determine the acceleration at t^{n+1}:

$$\tilde{\mathbf{M}}_s \mathbf{a}_s^{n+1} = \mathbf{F}_s^{n+1} - \mathbf{K}_s \mathbf{d}_s^p \tag{2.36}$$

In the last stage, displacements and velocities are computed as:

$$\mathbf{d}_s^{n+1} = \mathbf{d}_s^p + \frac{1}{4}\Delta t_s^2 \mathbf{a}_s^{n+1} \tag{2.37}$$

$$\mathbf{v}_s^{n+1} = \mathbf{v}_s^p + \frac{1}{2}\Delta t_s \mathbf{a}_s^{n+1} \tag{2.38}$$

2.1.2 Fluid sub-domain

Thereafter, the same development is made for the fluid sub-domain, in term of the equations used to describe the fluid environment and their numerical methods of resolution. The fluid sub-domain, called Ω_f, is considered alone. Initially, as in classical fluid problems, its formulation is Eulerian (see section 1.2.2); the frame used is fixed, with coordinate x. It is studied from the initial time $t^0 = 0$ until the final time $t^f = T$.

Continuous problem

The studied Newtonian fluid is considered compressible and inviscid. Moreover the boundaries are adiabatic; no heat is exchanged with the physical environments, as example between the fluid and the solid sub-domains. Under these assumptions, it is modeled by the Euler equations [37], which are composed of the balance of mass, momentum and energy. We choose to write them in their conservative form. The admissible conservative variables including (density ρ_f, momentum $\rho_f \underline{\mathbf{v}}_f$ and total volumic energy E_f) are searched over the continuous space $\Omega_f \times [0, T]$ such as:

$$\begin{cases} \frac{\partial \rho_f(\underline{\mathbf{x}},t)}{\partial t} + \nabla \cdot (\rho_f(\underline{\mathbf{x}},t)\underline{\mathbf{v}}_f(\underline{\mathbf{x}},t)) = 0 \\ \frac{\partial \rho_f(\underline{\mathbf{x}},t)\underline{\mathbf{v}}_f(\underline{\mathbf{x}},t)}{\partial t} + \underline{\nabla} \cdot (\rho_f(\underline{\mathbf{x}},t)\underline{\mathbf{v}}_f(\underline{\mathbf{x}},t) \otimes \underline{\mathbf{v}}_f(\underline{\mathbf{x}},t) + p_f(\underline{\mathbf{x}},t)\underline{\underline{\mathbf{I}}}) = \underline{\mathbf{0}} \\ \frac{\partial E_f(\underline{\mathbf{x}},t)}{\partial t} + \nabla \cdot ((E_f(\underline{\mathbf{x}},t) + p_f(\underline{\mathbf{x}},t))\underline{\mathbf{v}}_f(\underline{\mathbf{x}},t)) = 0 \end{cases} \tag{2.39}$$

From now, the space time dependency is skipped, but let's keep in mind that the frame used is Eulerian at this time.
The total volumic energy E_f is defined for a compressible flow as:

$$E_f = \frac{1}{2}\rho_f \underline{\mathbf{v}}_f^2 - \rho_f e \tag{2.40}$$

The internal energy e (per unit mass) can be interpreted as the energy associated with the random translation and internal motion of molecules plus the energy of interaction between them. Therefore, the internal energy is temperature and density dependent.
Finally, Euler equations are not a complete set of equations and require some additional constraints to admit a unique solution: these are the equations of state of the material considered. Here, a perfect gas is considered [26], whose pressure is defined by the perfect gas law as:

$$p_f = \rho_f R T_f = (k_f - 1)(E_f - \frac{1}{2}\rho_f \underline{\mathbf{v}}_f^2) \tag{2.41}$$

Where k_f is the specific heat ratio, R the individual gas constant and T_f the fluid temperature.
Adding a viscosity coefficient μ_f to the fluid, the equations (2.39) become the well-known Navier-Stokes equations [37] defined as:

$$\begin{cases} \frac{\partial \rho_f \underline{\mathbf{v}}_f}{\partial t} + \underline{\nabla} \cdot (\rho_f \underline{\mathbf{v}}_f \otimes \underline{\mathbf{v}}_f - \underline{\underline{\sigma}} f) = \underline{\mathbf{0}} \\ \nabla \cdot \underline{\mathbf{v}}_f = 0 \end{cases} \tag{2.42}$$

where the stress tensor is written as:

$$\underline{\underline{\sigma}}_f = -p_f \underline{\underline{\mathbf{I}}} + 2\mu_f \underline{\underline{\mathbf{D}}} \tag{2.43}$$

with the strain rate tensor $\underline{\underline{\mathbf{D}}}$ defined by:

$$\underline{\underline{\mathbf{D}}} = \frac{1}{2}(\underline{\nabla}\,\underline{\mathbf{v}}_f + \underline{\nabla}\,\underline{\mathbf{v}}_f^T) \tag{2.44}$$

Let's go back to the Euler equations, with $\mu_f = 0$. To be more readable, they are re-written using their vector form. The vector of conservative variables $\mathbf{U}_f$ and the flux vector $\mathbf{F}_f$ are defined below:

$$\mathbf{U}_f = \begin{bmatrix} \rho_f \\ \rho_f \underline{\mathbf{v}}_f \\ E_f \end{bmatrix} \qquad \mathbf{F}_f = \mathbf{F}_f(\mathbf{U}_f) = \begin{bmatrix} \rho_f \underline{\mathbf{v}}_f \\ \rho_f \underline{\mathbf{v}}_f \otimes \underline{\mathbf{v}}_f + p_f \underline{\underline{\mathbf{I}}} \\ (E_f + p_f)\underline{\mathbf{v}}_f \end{bmatrix} \tag{2.45}$$

The tensor notation is omitted to simplify this vector form. Then Euler equations (2.39) are re-written as:

$$\frac{\partial \mathbf{U}_f}{\partial t}|_{\mathbf{x}} + \nabla_{\mathbf{x}} \cdot \mathbf{F}_f = \mathbf{0} \tag{2.46}$$

As said in the section 1.2, an interface tracking approach is chosen in order to easier

ensure zero interface energy in the fluid-structure coupling. That is why, an ALE formulation (see section 1.2.2) is used on the fluid sub-domain. Hence, the fluid problem has to be re-written in an ALE frame, defined by the system of coordinates ξ which is time dependent. The velocity of the ALE grid is denoted by $\mathbf{w} = \frac{\partial \mathbf{x}}{\partial t}|_\xi$. It is Lagrangian at the boundaries and arbitrary inside the domain. This grid is different from the Eulerian grid, usually used for fluid problems, which remains fixed according to time. Moreover, it is also different from the Lagrangian grid used for the solid sub-domain where the velocity of the grid is the material velocity. Letus call $\mathbf{J}$ the jacobian of the frame transformation as $\mathbf{J} = \det \frac{\partial \mathbf{x}}{\partial \xi}|_t$.
The aim is now to rewrite the equation (2.46) in ALE formulation. First, the time derivation at constant coordinate ξ of the state vector is considered.

$$\frac{\partial \mathbf{U}_f}{\partial t}|_\xi = \frac{\partial \mathbf{U}_f}{\partial t}|_\mathbf{x} + \nabla_\mathbf{x} \mathbf{U}_f \cdot \frac{\partial \mathbf{x}}{\partial t}|_\xi \tag{2.47}$$

Using the definition of the ALE velocity and multiplying the equation (2.47) by $\mathbf{J}$ this one becomes:

$$\mathbf{J}\frac{\partial \mathbf{U}_f}{\partial t}|_\xi = \mathbf{J}\frac{\partial \mathbf{U}_f}{\partial t}|_\mathbf{x} + \mathbf{J}\nabla_\mathbf{x} \mathbf{U}_f \cdot \mathbf{w} \tag{2.48}$$

The jacobian property is recalled as:

$$\frac{\partial \mathbf{J}}{\partial t}|_\xi = \mathbf{J}\nabla_\mathbf{x} \cdot \mathbf{w} \tag{2.49}$$

Multiplying (2.49) by $\mathbf{U}_f$, and adding it to (2.48) we get:

$$\mathbf{J}\frac{\partial \mathbf{U}_f}{\partial t}|_\xi + \mathbf{U}_f\frac{\partial \mathbf{J}}{\partial t}|_\xi = \mathbf{J}\frac{\partial \mathbf{U}_f}{\partial t}|_\mathbf{x} + \mathbf{J}\nabla_\mathbf{x} \mathbf{U}_f \cdot \mathbf{w} + \mathbf{J}\mathbf{U}_f\nabla_\mathbf{x} \cdot \mathbf{w} \tag{2.50}$$

Then the first member of the equation (2.50) is re-written such as:

$$\frac{\partial \mathbf{J}\mathbf{U}_f}{\partial t}|_\xi = \mathbf{J}\frac{\partial \mathbf{U}_f}{\partial t}|_\mathbf{x} + \mathbf{J}\nabla_\mathbf{x} \mathbf{U}_f \cdot \mathbf{w} + \mathbf{J}\mathbf{U}_f\nabla_\mathbf{x} \cdot \mathbf{w} \tag{2.51}$$

Notice that:

$$\nabla_\mathbf{x} \cdot (\mathbf{U}_f \mathbf{w}) = \mathbf{U}_f\nabla_\mathbf{x} \cdot \mathbf{w} + \nabla_\mathbf{x} \mathbf{U}_f \cdot \mathbf{w} \tag{2.52}$$

Then the result of (2.52) is used into (2.51).

$$\frac{\partial \mathbf{J}\mathbf{U}_f}{\partial t}|_\xi = \mathbf{J}\frac{\partial \mathbf{U}_f}{\partial t}|_\mathbf{x} + \mathbf{J}\nabla_\mathbf{x} \cdot (\mathbf{U}_f \mathbf{w}) \tag{2.53}$$

Finally, the term $\frac{\partial \mathbf{U}_f}{\partial t}|_\mathbf{x}$ is replaced by its definition given by the Euler equations in eulerian formulation, equation (2.46).

$$\frac{\partial \mathbf{J}\mathbf{U}_f}{\partial t}|_\xi = -\mathbf{J}\nabla_\mathbf{x} \cdot \mathbf{F}_f + \mathbf{J}\nabla_\mathbf{x} \cdot (\mathbf{U}_f \mathbf{w}) \tag{2.54}$$

Thereafter the ALE flux vector is defined as:

$$\tilde{\mathbf{F}}_f = \mathbf{F}_f(\mathbf{U}_f) - \mathbf{w}\mathbf{U}_f \tag{2.55}$$

Reorganizing (2.54) and using the notation (2.55) the vector form of the Euler equations in ALE formulation is finally known as:

$$\frac{\partial \mathbf{J}\mathbf{U}_f}{\partial t}|_\xi + \mathbf{J}\nabla_\mathbf{x} \cdot \tilde{\mathbf{F}}_f = \mathbf{0} \tag{2.56}$$

Spatial dicretization

As for the solid sub-domain, in order to compute non-trivial solutions for the mathematical models derived, the differential equations have to be transformed into algebraic forms with a finite number of unknowns.
Concerning the spatial discretization, the choice is made in this work to use the common and powerful method for the fluid problems: the finite volume method. This one is based on the integral form of the conservative Euler equations. Thus, the equation (2.56) is integrated over a control volume called V_i.

$$\int_{V_i} \frac{\partial \mathbf{J}\mathbf{U}_f}{\partial t} \,\mathrm{d}\xi + \int_{V_i} \mathbf{J}\nabla_\mathbf{x}.\tilde{\mathbf{F}}_f \,\mathrm{d}\xi = \mathbf{0} \tag{2.57}$$

Let us remark that the controlled volume V_i is variable in time considering the Eulerian space, but in ALE coordinate it is time independent. Also, the partial time derivative in the previous equation (2.57) is evaluated at constant $\underline{\xi}$. In this way, it can be moved out of the integral.

$$\frac{d}{dt}\int_{V_i} \mathbf{J}\mathbf{U}_f \,\mathrm{d}\xi + \int_{V_i} \mathbf{J}\nabla_\mathbf{x} \cdot \tilde{\mathbf{F}}_f \,\mathrm{d}\xi = \mathbf{0} \tag{2.58}$$

Then, a variable change from ξ to $\mathbf{x}$, with $\mathrm{d}\mathbf{x} = \mathbf{J}\mathrm{d}\xi$ is operated to the equation (2.58) such as:

$$\frac{d}{dt}\int_{V_i} \mathbf{U}_f \,\mathrm{d}\mathbf{x} + \int_{V_i} \nabla_\mathbf{x} \cdot \tilde{\mathbf{F}}_f \,\mathrm{d}\mathbf{x} = \mathbf{0} \tag{2.59}$$

Finally, the divergence theorem is used, and the integral form of Euler equations in ALE formulation is written as:

$$\frac{d}{dt}\int_{V_i} \mathbf{U}_f \,\mathrm{d}\mathbf{x} + \int_{\Gamma_i} \tilde{\mathbf{F}}_f \cdot \mathbf{n} \,\mathrm{d}\mathbf{x} = \mathbf{0} \tag{2.60}$$

Then, the cell centered finite volume method is used (see figure 2.4). The idea is to divide the domain Ω_f into a finite number of control volumes V_i with $i = [1, \ldots, n_{cell}]$. n_{cell} is the number of finite volume as $\sum_{i=1}^{n_{cell}} V_i = \Omega_f$. Then the equation (2.60) is integrated over each control volume replacing the continuous state vector $\mathbf{U}_f$ and the continuous ALE flux vector $\tilde{\mathbf{F}}_f$ by discrete approximations [51].

Let us define $\mathbf{U}_{f_i}$, the approximation of $\mathbf{U}_f$ on the control volume V_i. It is defined as the mean of $\mathbf{U}_f$ over the finite control V_i and is considered at the center of this one:

$$\int_{V_i} \mathbf{U}_f \,\mathrm{d}\mathbf{x} \approx \Delta V_i \mathbf{U}_{f_i} \tag{2.61}$$

Hence, the name of the discretization method is cell centered finite volume. Vertex centered finite volume also exists and could be suitable for our FSI problem. Indeed,

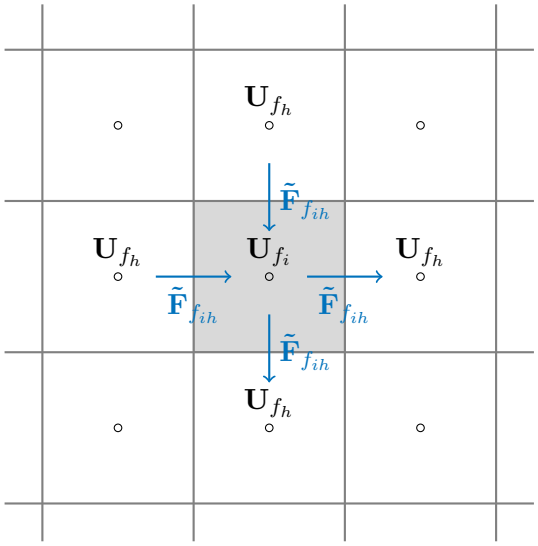


FIGURE 2.4: Cell centered finite volume discretization in 2D

unlike the cell centered method, in the vertex centered method, the discrete conservative variables vector is not defined at centers of the cells, but at their vertices, see Fig. 2.5. The value of fluid quantities, as velocities, are known directly at the boundaries, while they have to be approximated with the cell centered method. We will see later that in this work the coupling condition imposed to the fluid-structure interface is the velocity continuity. In this way, it would be more interesting to know accurately and easily these values at the boundaries. Nevertheless, the vertex centered method is harder to implement and less often used inside numerical solvers. For this reason, the adopted method is the finite volume cell centered method. To circumvent the problem of the unknown variable at the boundary, we use the ghost cells method. First, we define quantities at the cell faces as the mean between the neighbourhood cells of the considered face, illustrated by the black crosses on Fig. 2.5. Then, for the external boundaries, we add ghost cells, in dashed gray, and define the values of conservative variables at the center of those new cells as an extrapolation from the inner adjacent cells. In this way, values of quantities of interest are computed at the interface, exactly as for the other faces, as the mean between the adjacent ghost and inner cells.

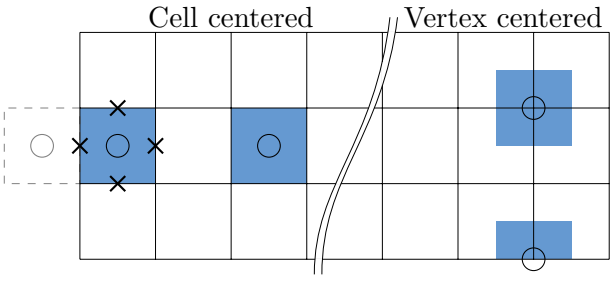


FIGURE 2.5: Finite volume discretization methods

Let us bring the digression about the vertex centered method to an end and go back to the finite volume cell centered method where we have to discretize the other term of the equation (2.60); the ALE flux vector. The flux is assumed as the sum of the flux across the boundaries of the control volume V_i, see Fig.2.4. Moreover, the flux

from the control volume V_h to the control volume V_i neighbour, is assumed constant on the boundary Γ_{ih} and is evaluated at its middle.

$$\int_{\Gamma_i} \tilde{\mathbf{F}}_f \cdot \mathbf{n} \, \mathrm{d}\mathbf{x} \approx \sum_{(\forall \Gamma_{ih} \in \Gamma_i)} s_{ih} \tilde{\mathbf{F}}_{f_{ih}} \tag{2.62}$$

Where s_{ih} is the area of the shared face between the control volume V_h and the control volume V_i. $\tilde{\mathbf{F}}_{f_{ih}}$ is called the numerical flux and is defined as $\tilde{\mathbf{F}}_{fih} = g(U_i, U_h, w_i, w_h)$. With g a chosen function, which should be conservative and consistent.

Hence, the Euler equations are written in their semi-discretized vector form as:

$$\frac{d}{dt}\big(\Delta V_i(t)\mathbf{U}_{f_i}(t)\big) + \sum_{\forall \Gamma_{ih} \in \Gamma_i} s_{ih}(t)\tilde{\mathbf{F}}_{f_{ih}}(t) = \mathbf{0} \qquad \forall i \in [1, ..., n_{cell}], \quad \forall t \in [0, T] \tag{2.63}$$

Time discretization

The next step consists now in the time discretization of the fluid sub-domain, as it has been done over the solid sub-domain. The idea is the same; we define an interval, called time step, and written Δt_f, such as $t^{n+1} = t^n + \Delta t_f$. Then the semi-discretized equation (2.63) is solved only at discrete times.

First, we approximate the time derivative as:

$$\frac{d}{dt}\big(\Delta V_i(t)\mathbf{U}_{f_i}(t)\big) = \frac{\Delta V_i^{n+1}\mathbf{U}_{f_i}^{n+1} - \Delta V_i^{n}\mathbf{U}_{f_i}^{n}}{\Delta t_f} \tag{2.64}$$

Then the choice of the function g defining the numerical flux, determines the time integration scheme. If g is a function which depends on discrete conservative variables and ALE grid velocity at the instant t^n, $g(U_i^n, U_h^n, w_i^n, w_h^n)$, the scheme is explicit in time, while if it depends on those variables at the instants t^n and t^{n+1}, $g(U_i^n, U_h^n, w_i^n, w_h^n, U_i^{n+1}, U_h^{n+1}, w_i^{n+1}, w_h^{n+1})$, the scheme is said implicit in time.
Thus, the general discretized formulation of the Euler equations in ALE formulation is written as:

$$\mathbf{U}_{f_i}^{n+1} = \frac{\Delta V_i^n}{\Delta V_i^{n+1}}\mathbf{U}_{f_i}^n - \frac{\Delta t_f}{\Delta V_i^{n+1}} \sum_{\forall \Gamma_{ih} \in \Gamma_i} s_{ih}^{n+\alpha}\tilde{\mathbf{F}}_{f_{ih}}^{n+\alpha} \tag{2.65}$$

Where $\alpha = 0$ for an explicit scheme and $\alpha = \frac{1}{2}$ for an implicit one.

Here the time scheme used is the Runge-Kutta 2 scheme, [102, 68]. This one is a two-step scheme, which is explicit and second order accurate. Applying this scheme

to the equation (2.65), it becomes:

$$\mathbf{U}_{f_i}^{n+\frac{1}{2}} = \frac{\Delta V_i^n}{\Delta V_i^{n+\frac{1}{2}}}\mathbf{U}_{f_i}^n - \frac{\Delta t_f}{2\Delta V_i^{n+\frac{1}{2}}}\sum_{\forall \Gamma_{ih}\in\Gamma_i} s_{ih}^n \tilde{\mathbf{F}}_{f_{ih}}^n \tag{2.66a}$$

$$\mathbf{U}_{f_i}^{n+1} = \frac{\Delta V_i^n}{\Delta V_i^{n+1}}\mathbf{U}_{f_i}^n - \frac{\Delta t_f}{\Delta V_i^{n+1}}\sum_{\forall \Gamma_{ih}\in\Gamma_i} s_{ih}^{n+\frac{1}{2}} \tilde{\mathbf{F}}_{f_{ih}}^{n+\frac{1}{2}} \tag{2.66b}$$

The final step is now to define the function g of the numeric flux. To be more readable, the time indices would be omitted until the end of this section. Here, the Roe flux difference splitting method has been chosen [100]. It is a method based on an approximated Riemann solver which is well adapted for compressible fluid. Indeed, for compressible flows, fluid properties are not only transported by the flow, but also by the propagation of waves. This requires the flux interpolation to be stabilized based on transport that can occur in any direction.
First, let's define the Jacobian of the flux $\mathbf{A}$, such as $\mathbf{A} = \frac{\partial \mathbf{F}_f}{\partial \mathbf{U}_f}$. Then let's call, κ the eigenvalues of $\mathbf{A}$, $\mathbf{r}$ the eigenvectors of $\mathbf{A}$ and α the Riemann invariants. K is the size of the matrix $\mathbf{A}$ and physically meaning, it is the number of waves. Also, κ^K is the velocity of the K^{th} wave, while $\mathbf{r}^K$ and α^K are its direction and strength respectively. Here, due to the ALE formulation the grid velocity has also to be taken into account in the computation of the wave velocity, in this way κ is replaced by $\tilde{\kappa}$, such as $\tilde{\kappa}^K = \kappa^K - \mathbf{w}_{i+\frac{1}{2}}$.
Thus, the numerical flux, between cells V_i and V_h is written using the Roe flux different splitting method as:

$$\tilde{\mathbf{F}}_{f_{ih}} = \frac{1}{2}\big(\tilde{\mathbf{F}}_{f_h} + \tilde{\mathbf{F}}_{f_i}\big) - \frac{1}{2}\sum_{k=0}^{5}\alpha^k|\tilde{\kappa}^k|\mathbf{r}^k \tag{2.67}$$

In order to define these values, we need to define them at Γ_{ih} using the Roe average as following:

$$\begin{aligned}
\rho_{f_{ih}} &= \sqrt{\rho_{f_i}\rho_{f_h}} \\
\mathbf{v}_{f_{ih}} &= \frac{\sqrt{\rho_{f_i}}\mathbf{v}_{f_i} + \sqrt{\rho_{f_h}}\mathbf{v}_{f_h}}{\sqrt{\rho_{f_i}} + \sqrt{\rho_{f_h}}} \\
h_{f_{ih}} &= \frac{\sqrt{\rho_{f_i}}h_{f_i} + \sqrt{\rho_{f_h}}h_{f_h}}{\sqrt{\rho_{f_i}} + \sqrt{\rho_{f_h}}} \\
c_{f_{ih}} &= \sqrt{(k_f - 1)\Big(h_{f_{ih}} - \tfrac{1}{2}|\mathbf{v}_{f_{ih}}|^2\Big)}
\end{aligned} \tag{2.68}$$

Where $h_f = \frac{E_f + p_f}{\rho_f}$ is the enthalpy and $c_f = \sqrt{\frac{k_f p_f}{\rho_f}}$ the sound velocity.
Then, the eigenvalues κ are defined according to the Roe average of the flow velocity and sound velocity.

$$\begin{aligned}
\kappa^1 &= v_{fx_{ih}} - c_{ih} \\
\kappa^2 &= v_{fx_{ih}} \\
\kappa^3 &= v_{fx_{ih}} \\
\kappa^4 &= v_{fx_{ih}} \\
\kappa^5 &= v_{fx_{ih}} + c_{ih}
\end{aligned} \tag{2.69}$$

Also, the values of the right eigenvectors of $\mathbf{A}$ could be found below :

$$\mathbf{r}^1 = \begin{bmatrix} 1 \\ v_{fx_{ih}} - c_{f_{ih}} \\ v_{fy_{ih}} \\ v_{fz_{ih}} \\ h_{fx_{ih}} - v_{ih} c_{f_{ih}} \end{bmatrix} \qquad \mathbf{r}^2 = \begin{bmatrix} 0 \\ 0 \\ v_{fy_{ih}} \\ 0 \\ v_{fy_{ih}}^2 \end{bmatrix} \qquad \mathbf{r}^3 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ v_{fz_{ih}} \\ v_{fz_{ih}}^2 \end{bmatrix}$$

$$\mathbf{r}^4 = \begin{bmatrix} 1 \\ v_{fx_{ih}} \\ v_{fy_{ih}} \\ v_{fz_{ih}} \\ v_{f_{ih}}^2 \end{bmatrix} \qquad \mathbf{r}^5 = \begin{bmatrix} 1 \\ v_{fx_{ih}} + c_{f_{ih}} \\ v_{fy_{ih}} \\ v_{fz_{ih}} \\ h_{f_{ih}} + v_{fx_{ih}} c_{f_{ih}} \end{bmatrix} \tag{2.70}$$

Finally, the Riemann invariants α are defined as :

$$\begin{aligned} \alpha^1 &= \tfrac{1}{2c_{f_{ih}}}\big((p_{f_h} - p_{f_i}) - \rho_{f_{ih}} c_{f_{ih}} (v_{fx_h} - v_{fx_i})\big) \\ \alpha^2 &= \tfrac{1}{c_{f_{ih}}}\big(c_{f_{ih}}^2 (\rho_{f_h} - \rho_{f_i}) - (p_{f_h} - p_{f_i})\big) \\ \alpha^3 &= \rho_{f_{ih}} (v_{fy_h} - v_{fy_i}) \\ \alpha^4 &= \rho_{f_{ih}} (v_{fz_h} - v_{fz_i}) \\ \alpha^5 &= \tfrac{1}{2c_{f_{ih}}}\big((p_{f_h} - p_{f_i}) + \rho_{f_{ih}} c_{f_{ih}} (v_{fx_h} - v_{fx_i})\big) \end{aligned} \tag{2.71}$$

In this section, the physics of fluid and solid sub-domains have been described, and the numerical methods used have been detailed. To sum up, a linear elastic solid sub-domain is considered, discretized by the finite element method and an implicit Newmark scheme. On the other side, the fluid sub-domain, compressible and inviscid is discretized by the cell centered finite volume method in an ALE formulation with a Roe flux difference splitting and an explicit Runge-Kutta scheme. In this way, we have to couple a linear implicit problem and a non-linear explicit problem. Finally, the time scales used on each sub-domain can be incompatible, if $\Delta t_s \neq \Delta t_f$.

2.2 Proposed coupling method

In this section, the proposed coupling method is detailed. The main objective is to propose a stable method for code coupling. First of all, as the stability is sought, as explained in the previous chapter, the proposed method is based on a monolithic formulation. Then, the coupling method should be as accurate and efficient as possible. Regarding these objectives, the use of appropriate, effective and existing dedicated solvers for solid and fluid dynamic computation have been chosen. Thus, a co-simulation algorithm is used to solve the monolithic system of equations. The use of dedicated solvers allows us to employ heterogeneous discretization methods over each sub-domain. Here, for the spatial discretization, the finite volume method is used over the fluid sub-domain while the finite element method is used over the structural sub-domain. Concerning the time discretization, fluid and solid solvers use explicit and implicit schemes respectively, as detailed in section 2.1. Moreover,

non-conforming interfaces can be used, in space, as detailed later in section 3.1.1, and also in time, with two different time scales.

2.2.1 Monolithic formulation

First, the monolithic formulation of the global coupled problem is presented. We consider an FSI problem without overlap, as described in the section 1.2.1.
The first step is to introduce the continuous variational formulation in time over each sub-domain, based on action integral. Over the solid sub-domain Ω_s, the subscript k is replaced by s respectively and f on the fluid sub-domain Ω_f:

$$\mathscr{A}_{\Omega_k} = \int_0^T \mathscr{L}_{\Omega_k} \, \mathrm{d}t \qquad k \in [s, f] \tag{2.72}$$

With the Lagrangian expressed as:

$$\mathscr{L}_{\Omega_k} = \mathscr{K}_{\Omega_k} - \left(\mathscr{W}_{int_{\Omega_k}} - \mathscr{W}_{ext_{\Omega_k}}\right) \qquad k \in [s, f] \tag{2.73}$$

Where $\mathscr{K}$ is the kinetic energy and $\left(\mathscr{W}_{int} - \mathscr{W}_{ext}\right)$ the potential energy, that is the difference between the internal and the external energy.
The equilibrium equations of each sub-domain are obtained by stationarization of these functionals, which is the Hamilton principle, defined as:

$$\delta\mathscr{A}_{\Omega_k} = 0 \qquad k \in [s, f] \tag{2.74}$$

The fluid-structure problem is then defined as the stationarization of equation (2.72) over each sub-domain. Moreover, coupling conditions are added to ensure fluid-structure interaction: the normal velocities continuity and the equilibrium condition of the interface, due to the inviscid fluid hypothesis:

$$\delta\mathscr{A}_{\Omega_s} = 0 \qquad \forall(\underline{\mathbf{X}}, t) \in \Omega_s \times [0, T] \tag{2.75a}$$

$$\delta\mathscr{A}_{\Omega_f} = 0 \qquad \forall(\underline{\xi}, t) \in \Omega_s \times [0, T] \tag{2.75b}$$

$$(\underline{\underline{\sigma}}_s \cdot \underline{\mathbf{n}}_s)\underline{\mathbf{n}}_s = p_f \qquad \forall(\underline{\mathbf{X}}, t) \text{ or } (\underline{\xi}, t) \in \Gamma_{FSI} \times [0, T] \tag{2.75c}$$

$$(\underline{\mathbf{n}}_s \cdot \underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s = (\underline{\mathbf{n}}_f \cdot \underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f \qquad \forall(\underline{\mathbf{X}}, t) \text{ or } (\underline{\xi}, t) \in \Gamma_{FSI} \times [0, T] \tag{2.75d}$$

Equations (2.75) can be viewed as a partitioned formulation of the FSI problem. On one hand, the resolution of equation (2.75a) using equation (2.75c) as a Neumann boundary condition, allows us to know the solid sub-domain behaviour. On the other hand, the equation (2.75b) is solved with (2.75d) as Dirichlet boundary condition to compute the behaviour of the fluid sub-domain. Each sub-system of equations can be solved independently only using the velocity, respectively pressure, as boundary conditions. As said in subsection 1.1.2, this kind of coupling gives some advantages, but suffers from a lack of accuracy and have no proof of stability in the general case.

Thus, our objective being to propose a stable coupling method, the next step is to move the partitioned system of equations (2.75) to a monolithic formulation. In this formulation, the entire FSI problem is thought globally, contrary to the partitioned

formulation. Thus, the variational functional of the FSI problem is written as the sum of the functional of each sub-domain, fluid and structure.

$$\delta\mathscr{A}_\Omega = \delta\Big(\mathscr{A}_{\Omega_s} + \mathscr{A}_{\Omega_f}\Big) = 0 \tag{2.76a}$$

$$(\underline{\underline{\sigma}}_s \cdot \underline{\mathbf{n}}_s)\underline{\mathbf{n}}_s = p_f \tag{2.76b}$$

$$(\underline{\mathbf{n}}_s \cdot \underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s = (\underline{\mathbf{n}}_f \cdot \underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f \tag{2.76c}$$

Then, to ensure the coupling condition of normal velocities continuity, the choice is made to transform the stationarization primal problem under constraints towards a dual problem by the mean of the Lagrange multipliers method. Thus, the dual Schur formulation of the functional of the FSI problem is written as:

$$\delta\mathscr{A}_\Omega \,\mathrm{d}t = \delta\Big(\mathscr{A}_{\Omega_s} + \mathscr{A}_{\Omega_f}\Big) + \delta\int_0^T \mathscr{L}_{\Gamma_{FS}}\, dt \tag{2.77}$$

The equation (2.77) can be viewed as the monolithic formulation of the FSI problem. The last term, $\mathscr{L}_{\Gamma_{FS}}$, is the variational energy of the interface. This one imposes the normal velocity continuity through the interface. Indeed, imposing this condition, rather than displacements continuity as example, guarantees the condition of stability for a general case as it has been proposed in [27]. Moreover, we recall that the Lagrangian frame is used over the solid sub-domain and the ALE frame is used for the fluid sub-domain. Both of these frames consider the boundaries as Lagrangian. That is why, on the fluid-structure interface Γ_{FS}, both frame coordinate systems are equivalent, $\underline{\mathbf{X}} = \underline{\xi}$, and they can be used without distinction. Then, the variational energy of the interface between fluid and solid sub-domains is rewritten in Lagrangian coordinates as:

$$\mathscr{L}_{\Gamma_{FS}} = \int_{\Gamma_{FS}} \underline{\Lambda} \cdot \big((\underline{\mathbf{n}}_s \cdot \underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f \cdot \underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X \tag{2.78}$$

Thanks to the Hamilton principle, the solid variational functional $\mathscr{L}_{\Omega_s}$ and the fluid variational functional $\mathscr{L}_{\Omega_f}$ stationarization gives the corresponding weak form, for all $\delta\underline{\mathbf{X}}_s$ and $\delta\underline{\mathbf{X}}_f$ respectively.
Thus the weak form of the solid equilibrium over Ω_s is written as:

$$\delta\mathscr{A}_{\Omega_s} = \int_0^T \Big[\int_{\Omega_1} \big[-\rho_s\underline{\mathbf{a}}_s \cdot \delta\underline{\mathbf{X}}_s - \underline{\underline{\underline{\mathbf{K}}}} : \underline{\underline{\epsilon}}(\underline{\mathbf{d}}_s) : \underline{\underline{\epsilon}}(\delta\underline{\mathbf{X}}_s) + \underline{\mathbf{f}}_s \cdot \delta\underline{\mathbf{X}}_s\big]\,\mathrm{d}X + \int_{\Gamma_{sn}} \big[\underline{\mathbf{f}}_{ext} \cdot \delta\underline{\mathbf{X}}_s\big]\,\mathrm{d}X\Big]\mathrm{d}t \tag{2.79}$$

In the same way, the momentum conservation over the fluid sub-domain Ω_f is written in its weak form as:

$$\delta\mathscr{A}_{\Omega_f} = \int_0^t \int_{\Omega_f} \delta\underline{\mathbf{X}}_f \cdot \big(\frac{\partial\rho_f\underline{\mathbf{v}}_f}{\partial t} + \underline{\nabla}.(\rho_f\underline{\mathbf{v}}_f \otimes \underline{\mathbf{v}}_f + p_f\underline{\underline{\mathbf{I}}})\big)\,\mathrm{d}x\mathrm{d}t \tag{2.80}$$

Let's now detail the stationarization of the interface functional. The first step is to develop the expression as:

$$\begin{aligned}\int_0^T \delta\mathscr{L}_{\Gamma_{FS}}\,\mathrm{d}t &= \int_0^T \delta \int_{\Gamma_{FS}} \underline{\Lambda}\big((\underline{\mathbf{n}}_s.\underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f.\underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\mathrm{d}t \\ &= \int_0^T \int_{\Gamma_{FS}} \delta\underline{\Lambda}\cdot\big((\underline{\mathbf{n}}_s\cdot\underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f\cdot\underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\mathrm{d}t \\ &\qquad + \int_0^T \int_{\Gamma_{FS}} \underline{\Lambda}\cdot\big((\underline{\mathbf{n}}_s\cdot\delta\underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f\cdot\delta\underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\mathrm{d}t\end{aligned}$$

Then applying integration by part on the second term of the development, the equation (2.2.1) becomes:

$$\begin{aligned}\int_0^T \delta\mathscr{L}_{\Gamma_{FS}}\,\mathrm{d}t &= \int_0^T \int_{\Gamma_{FS}} \delta\underline{\Lambda}\cdot\big((\underline{\mathbf{n}}_s\cdot\underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f\cdot\underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\mathrm{d}t \\ &\quad + \Big[\int_{\Gamma_{FS}} \underline{\Lambda}\cdot\big((\underline{\mathbf{n}}_s\cdot\delta\underline{\mathbf{X}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f\cdot\delta\underline{\mathbf{X}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\Big]_0^T \\ &\quad - \int_0^T \int_{\Gamma_{FS}} \dot{\underline{\Lambda}}\cdot\big((\underline{\mathbf{n}}_s\cdot\delta\underline{\mathbf{X}}_s)\underline{\mathbf{n}}_s - (\underline{\mathbf{n}}_f\cdot\delta\underline{\mathbf{X}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\mathrm{d}t \quad (2.81)\end{aligned}$$

Remaining that the virtual variables $\delta\underline{\mathbf{X}}_s$ and $\delta\underline{\mathbf{X}}_f$ are defined on $\mathscr{V}^0$ such as $\mathscr{V}^0 = \{\delta\underline{\mathbf{X}}_k \,|\, \delta\underline{\mathbf{X}}_k \in \mathscr{H}^1,\ \delta\underline{\mathbf{X}}_k(\underline{\mathbf{X}},t) = 0 \text{ on } \Gamma_D\times[0,T], \delta\underline{\mathbf{X}}_k(\underline{\mathbf{X}},0) = 0 \text{ on } \Omega_k, \delta\underline{\mathbf{X}}_k(\underline{\mathbf{X}},T) = 0 \text{ on } \Omega_k\}$ where k is the sub-domain subscript. Then, at the boundaries of the time integral, the virtual variables are equal to zero. Hence, the second term of the equation (2.81) is deleted.
Finally, the last term is developed and split into two time integrals, one for each virtual variables of both sub-domains. Hence, the expression of the interface functional is expressed as:

$$\begin{aligned}\int_0^T \delta\mathscr{L}_{\Gamma_{FS}}\,\mathrm{d}t &= \int_0^T \int_{\Gamma_{FS}} \delta\underline{\Lambda}\cdot\big((\underline{\mathbf{n}}_s\cdot\underline{\mathbf{v}}_s)\underline{\mathbf{n}}_s + (\underline{\mathbf{n}}_f\cdot\underline{\mathbf{v}}_f)\underline{\mathbf{n}}_f\big)\,\mathrm{d}X\mathrm{d}t \\ &\quad - \int_0^T \int_{\Gamma_{FS}} \dot{\underline{\Lambda}}\cdot(\underline{\mathbf{n}}_s\cdot\delta\underline{\mathbf{X}}_s)\underline{\mathbf{n}}_s\,\mathrm{d}X\mathrm{d}t \\ &\quad + \int_0^T \int_{\Gamma_{FS}} \dot{\underline{\Lambda}}\cdot(\underline{\mathbf{n}}_f\cdot\delta\underline{\mathbf{X}}_f)\underline{\mathbf{n}}_f\,\mathrm{d}X\mathrm{d}t \quad (2.82)\end{aligned}$$

By including equations (2.79), (2.80), (2.82) into equation (2.77), we get the global continuous monolithic equation of the FSI considered problem. Then, the idea is to split it into three equations, each dependent on one of virtual field, $\delta\underline{\mathbf{X}}_s$, $\delta\underline{\mathbf{X}}_f$ and $\delta\underline{\Lambda}$. These three equations are respectively the weak formulation of the momentum conservation over structural sub-domain and over fluid sub-domain as well as the coupling condition at the interface.
Finally, those three equations are discretized in space using the discretization methods described in the previous section 2.1. The semi discretized equation of solid sub-domain in Lagrangian formulation is directly obtained as given in the equation (2.83a). The calculation of the semi-discretization in finite volume with ALE formulation from the weak formulation of the fluid momentum conservation, is less straightforward. The detailed calculation is presented into the Appendix A, and gives equation (2.83b) for the cell i. The last one semi-discretized equation (2.83c), represents the continuity of

the nodal normal velocities at the interface.

$$\mathbf{M}_s\mathbf{a}_s + \mathbf{K}_s\mathbf{d}_s = \mathbf{F}_s + \mathbf{L}_s^T\mathbf{\Lambda} \qquad \text{in } \Omega_s \tag{2.83a}$$

$$\frac{d}{dt}\left(\Delta V_i \mathbf{U}_{f_i}\right) = -\sum_{\forall \Gamma_{ih} \in \Gamma_i} \left(s_{ih}\tilde{\mathbf{F}}_{f_{ih}} + \mathbf{L}_{f_{ih}}\mathbf{\Lambda}_{ih}\right) \qquad \text{in } \Omega_f \tag{2.83b}$$

$$\mathbf{L}_s\mathbf{v}_s\mathbf{n}_s = \mathbf{l}_f\mathbf{v}_{f_{vertex}}\mathbf{n}_f \qquad \text{on } \Gamma_{FS} \tag{2.83c}$$

Where $\mathbf{\Lambda}$ is the vector of the time discretized derived Lagrange multipliers. This one is homogeneous to a force and could be viewed as interaction forces between the two sub-domains.
We define $\mathbf{L}_s$, the normal node selection matrix of the solid mesh belonging to the interface Γ_{FS}. In the same way, $\mathbf{L}_f$ is a $(n_{vertex} \times 3)$ matrix, where the first and the last columns are zeros vectors and the second column is the transposed of the normal vertex selection matrix $\mathbf{l}_f$. The matrix $\mathbf{L}_f = [\mathbf{0}, \mathbf{l}_f^T, \mathbf{0}]$ allows us to take into account interaction forces in the momentum conservation equation, for the cells whose one or more faces are included in Γ_{FS}. In other words, interaction forces do not influence the mass and the energy conservation in the fluid computation. When the meshes are compatible; when the node of the fluid mesh and solid mesh are common on the interface, these operators are Boolean.
Finally, $\mathbf{v}_{f_{vertex}}$ is the vector of fluid velocities at the vertices. Since we used cell centered finite volume method as spatial discretization, fluid velocities are defined at cell centers. Nevertheless, the interface functional imposes normal velocity continuity on the interface Γ_{FS}. Then, for the semi discrete form, we introduced the vector $\mathbf{v}_{f_{vertex}}$. This one could be built using different kind of approximation, such as gradient or ghost cells and will be detailed for each test case.

Concerning the time discretization and for the sake of the generalization, we choose to use a specific scale on each sub-domain. In view of hypotheses and time discretization schemes, the fluid sub-domain is discretized using a micro time scale while the solid sub-domain is discretized using a macro time scale. The explicit Runge-Kutta scheme used over fluid sub-domain is conditionally stable according a critical time step, while the implicit Newmark scheme used for structural computation, is unconditionally stable. Thus Δt_f is the micro time step between the times t^{j-1} and t^j, and Δt_s is the macro-time step between the instants t^0 and t^m such as $\Delta t_s = m\Delta t_f$, with $m \in \mathbb{N}^*$. In this work, we consider that the fluid and solid simulation start at the same time and that the ratio between the two scales m is an integer. In this way, every m time step both sub-domains are computed for the same time, as it is shown on Fig. 2.6 where $m = 3$.
About the time discretization of the coupling condition at the interface, equation (2.83c), two options are available. It could be done at the micro time scale or at the larger one. The main advantage of the coupling at the macro scale is that it is conservative in term of energy, but the interface tracking on the explicit micro scale is difficult and less accurate. On the other way, when the coupling condition is imposed at the micro time step, the geometry motions of the interface is well known but this method induces a small amount of dissipative energy [52]. Nevertheless, the loss of energy is often considered small enough compared to the global energy balance, as in [53, 95, 19]. Last but not least, the coupling at the fine time scale can be implemented really more easily than at the macro scale. That is why, in this work the micro time-step coupling has been chosen.

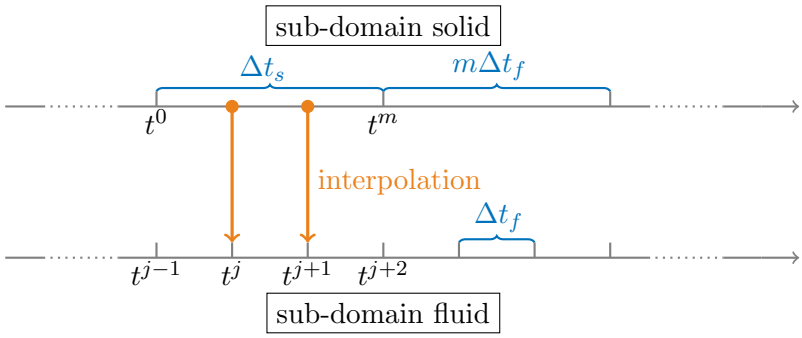


FIGURE 2.6: Fluid and solid time scales with $m = 3$

Thus, the normal velocity continuity condition through the Fluid-Structure interface is discretized at the fine time scale, in our case the fluid time scale. The coupling condition is written at the Runge-Kutta mid-step and at the micro time step. Finally the discretized equations of the global coupled problem, viewed as monolithic, are written as:

$$\begin{cases} \tilde{\mathbf{M}}_s \mathbf{a}_s^m = \mathbf{F}_s^m - \mathbf{K}_s \mathbf{d}_s^P + \mathbf{L}_s^T \mathbf{\Lambda}^m \\ \Delta V_{f_i}^{j-\frac{1}{2}} \mathbf{U}_{f_i}^{j-\frac{1}{2}} = \Delta V_{f_i}^{j-1} \mathbf{U}_{f_i}^{j-1} - \frac{\Delta t_f}{2} \sum \left(s_{f_{ih}}^{j-1} \tilde{\mathbf{F}}_{f_{ih}}^{j-1} + \mathbf{L}_{f_{ih}}^T \mathbf{\Lambda}_{ih}^{j-\frac{1}{2}} \right) \\ \Delta V_{f_i}^{j} \mathbf{U}_{f_i}^{j} = \Delta V_{f_i}^{j-1} \mathbf{U}_{f_i}^{j-1} - \Delta t_f \sum \left(s_{f_{ih}}^{j-\frac{1}{2}} \tilde{\mathbf{F}}_{f_{ih}}^{j-\frac{1}{2}} + \mathbf{L}_{f_{ih}}^T \mathbf{\Lambda}_{ih}^{j} \right) \\ \mathbf{L}_s \mathbf{v}_s^{j-\frac{1}{2}} \mathbf{n}_s^{j-\frac{1}{2}} = \mathbf{l}_f \mathbf{v}_{f_{vertex}}^{j-\frac{1}{2}} \mathbf{n}_f^{\frac{1}{2}} \\ \mathbf{L}_s \mathbf{v}_s^{j} \mathbf{n}_s^{j} = \mathbf{l}_f \mathbf{v}_{f_{vertex}}^{j} \mathbf{n}_f^{j} \end{cases} \tag{2.84}$$

2.2.2 Co-simulation algorithm

The objective is now to solve the monolithic discretized system of equations (2.84) governing the FSI problem into a co-simulation algorithm. The idea of the co-simulation algorithm is to allow the equations of each sub-domain to be solved by dedicated CFD solver and solid dynamic solver.

As said previously, we made the choice to impose the coupling condition, the normal velocities continuity, at the fine time scale, which is the fluid time scale. Moreover, the fluid time discretization uses the two-step Runge-Kutta scheme. In this way, the velocity continuity has also to be imposed at this Runge-Kutta mid step. Thus, the first step is to define the solid velocities at the micro time scale and at the Runge-Kutta mid-step, in order to solve the coupling equation. A simple linear interpolation is used, as proposed by the GC method:

$$\mathbf{v}_s^{j-\frac{1}{2}} = (1 - \frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0 + \frac{j-\frac{1}{2}}{m}\mathbf{v}_s^m \tag{2.85a}$$

$$\mathbf{v}_s^{j} = (1 - \frac{j}{m})\mathbf{v}_s^0 + \frac{j}{m}\mathbf{v}_s^m \tag{2.85b}$$

The same kind of interpolation will be needed for the itnteraction forces $\mathbf{\Lambda}$. Then they are defined as:

$$\mathbf{\Lambda}^{j-\frac{1}{2}} = (1 - \frac{j-\frac{1}{2}}{m})\mathbf{\Lambda}^0 + \frac{j-\frac{1}{2}}{m}\mathbf{\Lambda}^m \tag{2.86a}$$

$$\mathbf{\Lambda}^{j} = (1 - \frac{j}{m})\mathbf{\Lambda}^s 0 + \frac{j}{m}\mathbf{\Lambda}^m \tag{2.86b}$$

The idea of the co-simulation algorithm is based on a predictor-corrector decomposition. First, the variables called "*free*" are defined. They are the solid state vector and the fluid conservative variables vector describing the behaviour of the solid sub-domain and fluid-sub-domain respectively without taking into account the interface interactions at the current time step. In other words, *free* variables depend on the materials history, but do not depend on $\mathbf{\Lambda}^m$, $\mathbf{\Lambda}^{j-\frac{1}{2}}$ or $\mathbf{\Lambda}^j$. In this way, *free* state could be viewed as a predictive state. On the other side, we define the "*link*" variables which describe the behaviour of each sub-domain induced by the fluid-structure interaction forces alone. The sum of *free* states and the *link* states of each sub domain gets the global structural state vector and conservative fluid variables, such as:

$$\mathbf{U}_{k_{free}} + \mathbf{U}_{k_{link}} = \mathbf{U}_k \qquad k = s, f \tag{2.87}$$

Then, solving the system of equations (2.84), it appears that Lagrange multipliers only depend on *free* variables (see appendix B for more details). In this way, the resolution of the fluid-structure problem can be done by the general following procedure, summarized in Fig.2.7. First, the *"free"* states of each sub-domain are computed from the previous time step and boundary conditions (except for the coupling condition). Then, the interface operators are computed using the *free* velocities. Using those ones, the *link* states of fluid and solid sub-domains can be computed. And finally, the global state of the fluid-structure problem is known by the summation of *free* and *link* solid state vectors in the one hand and the *free* and *link* fluid conservative variables vectors in the other hand.

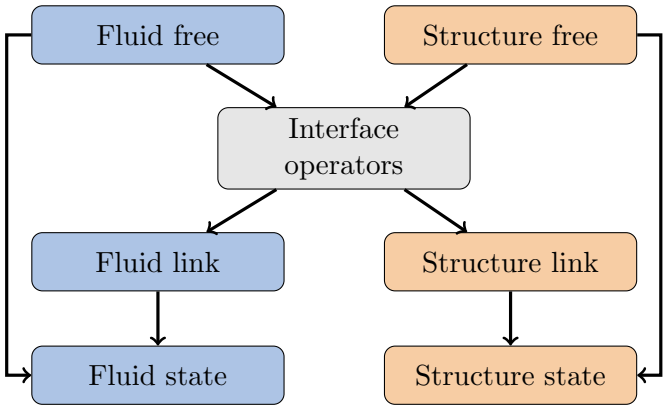


FIGURE 2.7: Free/Link decomposition for the co-simulation algorithm

Following, the equations solved by the proposed algorithm are detailed. First, let us split the discretized solid state vector into *free* variables and *link* variables. This split comes immediately such as the *free* accelerations, $\mathbf{a}^m_{s_{free}}$ are defined by the terms non-depending on the Lagrange multipliers Λ^m while *link* accelerations $\mathbf{a}^m_{s_{link}}$ are defined only by the term depending on the interface operators. Then, displacements

and velocities are defined in the same way. The *free* states are calculated from the Newmark predictions and the term depending on *free* accelerations, while the *link* states are calculated from the term depending on *link* accelerations.

$$\mathbf{a}^m_{s_{free}} = \tilde{\mathbf{M}}_s^{-1}\left(\mathbf{F}^m - \mathbf{K}_s\mathbf{d}_s^p\right) \quad (2.88a) \qquad \mathbf{a}^m_{s_{link}} = \tilde{\mathbf{M}}_s^{-1}\mathbf{L}_s^T\Lambda^m \quad (2.89a)$$

$$\mathbf{v}^m_{s_{free}} = \mathbf{v}_s^P + \frac{\Delta t_s}{2}\mathbf{a}^m_{s_{free}} \quad (2.88b) \qquad \mathbf{v}^m_{s_{link}} = \frac{\Delta t_s}{2}\mathbf{a}^m_{s_{link}} \quad (2.89b)$$

$$\mathbf{d}^m_{s_{free}} = \mathbf{d}_s^P + \frac{\Delta t_s^2}{4}\mathbf{a}^m_{s_{free}} \quad (2.88c) \qquad \mathbf{d}^m_{s_{link}} = \frac{\Delta t_s^2}{4}\mathbf{a}^m_{s_{link}} \quad (2.89c)$$

The same split has to be done over the fluid sub-domain Ω_f. This time, the operation is less obvious than over the solid sub domain. Indeed, into the fluid equations of the system (2.84), there are two kinds of variables which are dependent on interface operators: the momentum $\left(\rho\mathbf{v}\right)_f^{j-\frac{1}{2}}$ and $\left(\rho\mathbf{v}\right)_f^{j}$ and the cells volume $\Delta V^{j-\frac{1}{2}}$ and ΔV^j. That is why, instead of splitting the conservative variables vector alone, the cell volumes and conservative variables vector are grouped. It is this new vector of grouped variables that is split into *free* and into *link* states. Let's call it mean conservative variables vector, simply noted $\left(\Delta V\mathbf{U}_f\right)$. Thus, the *free* mean conservative variables vector at Runge-Kutta mid step $t^{j-\frac{1}{2}}$ and at micro time step t^j is computed by the mean conservative variables from the previous time step and by the explicit numerical flux:

$$\left(\Delta V\mathbf{U}_f\right)^{j-\frac{1}{2}}_{free_i} = \Delta V_i^{j-1}\mathbf{U}^{j-1}_{f_i} - \frac{\Delta t_f}{2}\sum s^{j-1}_{ih}\tilde{\mathbf{F}}^{j-1}_{f_{ih}} \quad (2.90a)$$

$$\left(\Delta V\mathbf{U}_f\right)^{j}_{free_i} = \Delta V_i^{j-1}\mathbf{U}^{j-1}_{f_i} - \Delta t_f\sum s^{j-\frac{1}{2}}_{ih}\tilde{\mathbf{F}}^{j-\frac{1}{2}}_{f_{ih}} \quad (2.90b)$$

On the other side, the *link* mean conservative variables vector is expressed by the means of the interface operator dependent term, such as:

$$\left(\Delta V\mathbf{U}_f\right)^{j-\frac{1}{2}}_{link_i} = -\frac{\Delta t}{2}\sum \mathbf{L}^T_{f_{ih}}\Lambda^{j-\frac{1}{2}}_{ih} \quad (2.91a)$$

$$\left(\Delta V\mathbf{U}_f\right)^{j}_{link_i} = -\Delta t\sum \mathbf{L}^T_{f_{ih}}\Lambda^{j}_{ih} \quad (2.91b)$$

Letus recall that the first and the last row of the fluid selection operator $\mathbf{L}_f$ are zeros vectors. In this way, the first row of the *link* mean conservative variables vector is zero, $\left(\Delta V\rho_f\right)_{link}$, as its last row, $\left(\Delta VE_f\right)_{link}$. Using this property, we can define the fluid, free and link velocities even if the cells volumes are unknown, such as:

$$\mathbf{v}_{f_{free}} = \frac{\left(\Delta V\rho_f\mathbf{v}_f\right)_{free}}{\left(\Delta V\rho_f\right)_{free}} \quad (2.92) \qquad \mathbf{v}_{f_{link}} = \frac{\left(\Delta V\rho_f\mathbf{v}_f\right)_{link}}{\left(\Delta V\rho_f\right)_{free}} \quad (2.93)$$

With these definitions, the interface operators can now be expressed. The equations (2.88b) and (2.92) are included into the equation of the coupling condition, the last equation of the system (2.84). Then, Runge-Kutta mid-step and micro-time step interpolations, equations (2.85) and (2.86) are used. The detailed calculations are available in Appendix B. Finally, interface operators at the Runge-Kutta mid step

$\mathbf{\Lambda}^{j-\frac{1}{2}}$ and at the micro time step Λ^j are defined as follows:

$$\left(\mathbf{H}_s + \mathbf{H}_f^{j-\frac{1}{2}}\right)\mathbf{\Lambda}^{j-\frac{1}{2}} = \mathbf{L}_s \mathbf{v}_{s_{free}}^{j-\frac{1}{2}} \mathbf{n}_s^{j-\frac{1}{2}} + \mathbf{l}_f \mathbf{v}_{vertex_{f_{free}}}^{j-\frac{1}{2}} \mathbf{n}_f^{j-\frac{1}{2}} \tag{2.94a}$$

$$\left(\mathbf{H}_s + \mathbf{H}_f^{j}\right)\mathbf{\Lambda}^{j} = \mathbf{L}_s \mathbf{v}_{s_{free}}^{j} \mathbf{n}_s^{j} + \mathbf{l}_f \mathbf{v}_{vertex_{f_{free}}}^{j} \mathbf{n}_f^{j} \tag{2.94b}$$

Where the Steklov Poincaré operators $\mathbf{H}$ for fluid and solid are defined as:

$$\mathbf{H}_s = \frac{1}{2}\Delta t_s \mathbf{L}_s \tilde{\mathbf{M}}_s t_f \mathbf{L}_s^T \tag{2.95a}$$

$$\mathbf{H}_f^{j-\frac{1}{2}} = \frac{1}{2}\Delta t_f \mathbf{l}_f \mathbf{M}_f^{j-\frac{1}{2}^{-1}} \mathbf{l}_f^T \tag{2.95b}$$

$$\mathbf{H}_f^{j} = \Delta t_f \mathbf{l}_f \mathbf{M}_f^{j^{-1}} \mathbf{l}_f^T \tag{2.95c}$$

With $\mathbf{M}_f$ the fluid mass matrix, defined as $\mathbf{M}_f = \left(\Delta V \rho_f\right)_{vertex}\mathbf{I}$. Recalling that $\left(\Delta V \rho_f\right)_{link}$ is zero, the fluid mass matrix is known by the fluid "free" computation. Let's remark that the Steklov Poincare operator for solid $\mathbf{H}_s$ is time independent, contrary to the fluid one $\mathbf{H}_f$. This is due to the assumption of compressible fluid. In this way, the solid operator can be computed only once at the initialization while the fluid operator should be updated at each Runge-Kutta mid-step and at each micro time step.

We now get all the keys to detail the monolithic co-simulation algorithm of the fluid-structure interaction problem considered. Fig 2.8 shows a representation of the proposed algorithm with a ratio $m = 2$ between the fluid and the solid time scales.

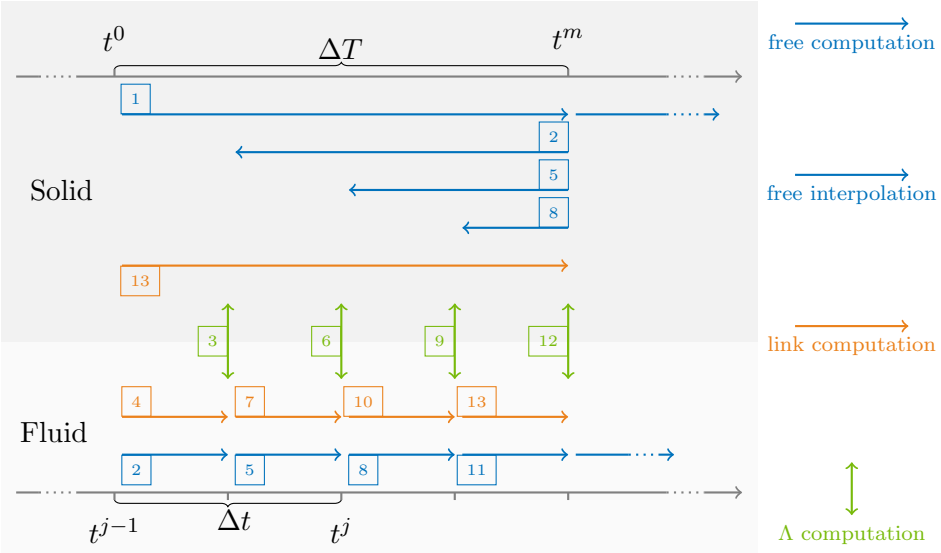


FIGURE 2.8: Multi time-step algorithm with $m = 2$

In the general case, the procedure is described below and by the Alg.2. First of all, let us begin with the initialization. Thanks to the initial conditions, the states of the fluid sub-domain Ω_f and of the solid sub-domain Ω_s are computed at $t^0 = 0$. Moreover the solid invariants, the modified mass matrix $\tilde{\mathbf{M}}_s$ and the Steklov-Poincare operator $\mathbf{H}_s$, are evaluated.

Following which the time loop is started over the macro time scale until the final step is achieved. At each macro-time step, the first step is to compute the *free* state vector of the solid sub-domain $\mathbf{U}^m_{s_{free}}$ according to equations (2.88).
Then the micro-time loop is run, from $j = 1$ to $j = m$. First, the Runge-Kutta mid-step is processed. The *free* structural velocities at the interface, are interpolated by equation (2.85a) and the fluid *free* mean conservative variables are computed by equation (2.90a). In order to solve the interface problem, the mean conservative *free* variables vector is extrapolated at Γ_{FS}. Then, the fluid operator $\mathbf{H}_f^{j-\frac{1}{2}}$ can be computed from equation (2.95b). Afterwards, the Lagrange multipliers are determined using equation (2.94a). Knowing interface operators, the mean conservative *link* variables vector is computed, equation (2.91a). Thus, using the equation (2.93), the relation (2.87) and the ghost cell method for the extrapolation at the interface, the fluid velocities at $\Gamma_{FSI}^{j-\frac{1}{2}}$ are deduced. According to it, the ALE grid is updated. Hence, the cells volumes $\Delta V^{j-\frac{1}{2}}$ are no longer unknown. Finally, adding the *free* and the *link* mean conservative variables vectors and divided it by the current cells volumes, we get the total fluid conservative variables vector at the Runge-Kutta mid-step.
Then, to know the fluid state at the end of the micro time step, exactly the same procedure is followed for the second Runge-Kutta step. Where the explicit ALE flux can be now computed since the grid velocities $\mathbf{w}^{j+\frac{1}{2}}$ have been computed by the previous grid update.
When the last micro time loop has been performed, $j = m$, we go out of the micro time loop to go back to the macro one, with the aim to end the computation of the structural state. Using the last Lagrange multiplier computed at the fine time step, the *link* state vector is computed, equation (2.89). Finally, using the equation (2.87) the total solid state is deduced at the end of the current macro time step. The macro-loop is re-launched until the final time step is reached, such as $t^m = T$.

In this section, the proposed method for an FSI coupling based on a monolithic formulation and solved by a co-simulation algorithm has been described. The objective of the method is to aim for stability, accuracy and efficiency in numerical simulation applied on FSI problems. Concerning the stability, the use of the monolithic formulation instead of partitioned formulation should forestall performance degradation. Moreover, stability is guaranteed by the coupling condition imposed synchronously on the two sub-domains, by normal velocities continuity at the interface. Also, the proposed method could be said heterogeneous which allows a better accuracy. Indeed, each sub-domain uses a different discretization method, adapted to its hypotheses. The solid sub-domain is discretized by the finite element method and an implicit Newmark scheme, while the fluid sub-domain uses the finite volume method in ALE formulation and an explicit Runge-Kutta scheme. Finally, to improve its efficiency, the method proposes to use dedicated time and space scale on each sub-domain. The meshes can be non-matching at the interface and the fluid sub-domain can be solved using a finer time step than the one used on the structural sub-domain.
The proposed method has now to be validated with academic test cases which will be presented is the next section.

Algorithm 2 Multi time step FSI coupling with MCS method

Initial fluid Ω_f and solid Ω_s states
Compute structural invariant : $\tilde{\mathbf{M}}_s, \mathbf{H}_s \leftarrow$ (2.95a)
while $t^m \leq T$ **do** ▷ Macro time loop
  Compute $\mathbf{U}^m_{s_{free}} \leftarrow$ (2.88)
  for $j \in [1, m]$ **do** ▷ Micro time loop
    Interpolate $\mathbf{v}^{j-\frac{1}{2}}_{s_{free}} \leftarrow$ (2.85a) ▷ Runge-Kutta first step
    Compute $\left(\Delta V \mathbf{U}\right)^{j-\frac{1}{2}}_{f_{free}} \leftarrow$ (2.90a)
    Extrapolate $\mathbf{l}_f \left(\Delta V \rho\right)^{j-\frac{1}{2}}_{f_{free}}$ and $\mathbf{l}_f \mathbf{v}^{j-\frac{1}{2}}_{f_{free}} \leftarrow$ (2.92)
    Compute $\mathbf{H}^{j-\frac{1}{2}}_f \leftarrow$ (2.95b)
    Compute $\mathbf{\Lambda}^{j-\frac{1}{2}} \leftarrow$ (2.94a)
    Compute $\left(\Delta V \mathbf{U}\right)^{j-\frac{1}{2}}_{f_{link}} \leftarrow$ (2.91a)
    Compute $\mathbf{l}_f \mathbf{v}^{j-\frac{1}{2}}_f \leftarrow$ (2.87) and (2.93)
    Compute $\Delta V^{j-\frac{1}{2}}, \mathbf{w}^{j-\frac{1}{2}} \leftarrow$ mesh update
    Compute $\mathbf{U}^{j-\frac{1}{2}}_f \leftarrow$ (2.87)
    Interpolate $\mathbf{v}^j_{s_{free}} \leftarrow$ (2.85b) ▷ Runge-Kutta second step
    Compute $\left(\Delta V \mathbf{U}\right)^j_{f_{free}} \leftarrow$ (2.90b)
    Extrapolate $\mathbf{l}_f \left(\Delta V \rho\right)^j_{f_{free}}$ and $\mathbf{l}_f \mathbf{v}^j_{f_{free}} \leftarrow$ (2.92)
    Compute $\mathbf{H}^j_f \leftarrow$ (2.95c)
    Compute $\mathbf{\Lambda}^j \leftarrow$ (2.94b)
    Compute $\left(\Delta V \mathbf{U}\right)^j_{f_{link}} \leftarrow$ (2.91b)
    Compute $\mathbf{l}_f \mathbf{v}^j_f \leftarrow$ (2.87) and (2.93)
    Compute $\Delta V^j, \mathbf{w}^j \leftarrow$ mesh update
    Compute $\mathbf{U}^j_f \leftarrow$ (2.87)
  end for
  compute $\mathbf{U}^m_{s_{link}} \leftarrow$ (2.92)
  compute $\mathbf{U}^m_s \leftarrow$ (2.87)
end while

2.3 Piston test case validation

The present section is dedicated to the validation of the proposed coupling method for the considered fluid-structure interaction problems. The choice was made to use the one dimensional piston test case which is a classical academic test case used to evaluate FSI coupling methods. The aim will be to evaluate the performance of the proposed method in terms of general results, error of coupling and energy preservation. The results will also be compared with a partitioned coupling approach.

2.3.1 Presentation of the problem

The piston test case has been introduced in 1995 by Piperno [93] whose aim was to compare several approaches for partitioned fluid-structure coupling.
We considered a one meter tube, where is enclosed air. The left end of the chamber is closed by a wall while the right end is closed by a mass-spring system, see Fig.2.9. This mass-spring system is the solid sub-domain considered and the tube is the fluid

sub-domain.

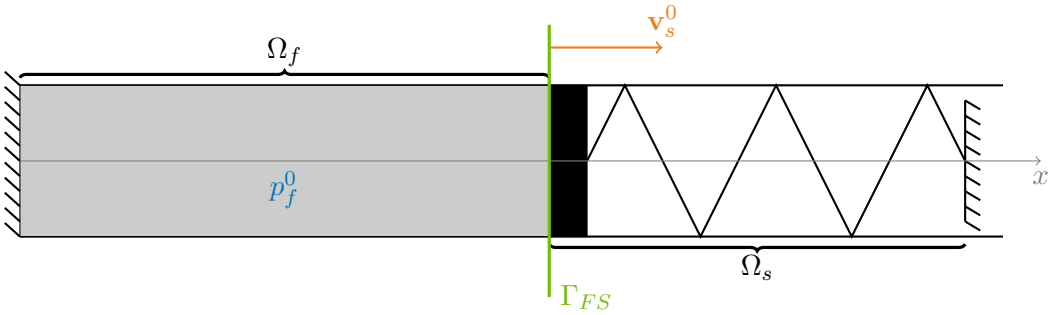


FIGURE 2.9: 1D piston problem, geometry, initial and boundaries conditions

This test case has been re-used latter to validate FSI coupling. As an example, Blom [16] proposed to couple the finite volume method with ALE formulation for fluid sub-domain to Lagrangian structure with a monolithic time integration algorithm with linear extrapolation for the ALE grid. Then, Michler [82] proposed to couple Galerkin finite element for fluid and solid sub-domains synchronously. Lefrançois [71] also used this test case to study different formulation for the fluid sub-domain, using finite element in eulerian, Lagrangian or ALE formulation. More recently, Ischinger [59] proposed a monolithic coupling with interface capturing and discontinuous Galerkin method on fluid sub-domain.
In all this work the materials' parameters and initial conditions vary a bit. In this work we chose to use the same set of parameters as the one used by Blom and Ischinger. Concerning the materials, the fluid sub-domain Ω_f is made of air, with a specific heat ratio $k_f = 1.4$ and initial density $\rho_f^0 = 1.3Kg.m^{-3}$. It is considered compressible and as a perfect gas. It is discretized by 100 finite volumes. The solid sub-domain Ω_s, is made of a linear mass-spring system undamped, where the mass is $m_s = 0.8Kg$ and the stiffness is $k_s = 8000N.m^{-1}$. It is considered to have a single degree of freedom. At the beginning of the simulation, the pressure of the fluid is the same as the atmospheric pressure outside the tube as $p_f^0 = 1e^5Pa$. Then an initial velocity is given to the mass, such as $v_s^0 = 20m.s^{-1}$. The set of parameters used is recapitulated into the Tab.2.1.

Geometry	symbol	value
Length of the fluid chamber at rest	l_f^0	$1\,\mathrm{m}$
Number of finite volumes of Ω_f	n_{cell}	100
Number of degrees of freedom of Ω_s	n_{dof}	1
Material	symbol	value
Specific heat ratio of Ω_f	k_f	1.4
Mass of the Ω_s mass	m_s	$0.8\,\mathrm{kg}$
Stiffness of the Ω_s spring	k_s	$8000\,\mathrm{N\,m^{-1}}$
Initial condition	symbol	value
Initial pressure of Ω_f	p_f^0	$1 \times 10^5\,\mathrm{Pa}$
Initial density of Ω_f	ρ_f^0	$1.3\,\mathrm{kg\,m^{-3}}$
Initial velocity of Γ_{FS}	$\mathbf{v}_s^0$	$20\,\mathrm{m\,s^{-1}}$

TABLE 2.1: Set of parameters for the 1D piston problem

As we said earlier, the explicit Runge-Kutta scheme used for the fluid computation is conditionally stable. In this way, the micro time step chosen to solve the fluid sub-domain shouldn't be inferior to the critical time step of the scheme. In this one dimensional test case with ALE formulation, the critical time step is known as:

$$\Delta t_{crit} = \frac{min(\Delta V)}{c_f + |\mathbf{v}_f - \mathbf{w}|} \tag{2.96}$$

As the grid is moving, the critical time step is time dependent. Here, we simply used a constant time step for each sub-domain. In this way, we have to use a fluid time step smaller than the worst critical time step. This one is defined by the smallest ALE grid configuration. Considering the behaviour of the coupling problem as unknown, we choose the most unfavourable configuration as the maximum position of the piston squeezed, when this one is computed without fluid interaction. Then we get $\Delta t_{crit} = 2.5 \times 10^{-5}\,\mathrm{m\,s^{-1}}$.

The last step to describe this test case is the way the ALE grid will be solved. Fig. 2.10 shows the configuration of the fluid mesh at two consecutive instants t^{j-1} and t^j. The grid motion is updated from the interface motion, conserving the same number of cells. Moreover, the cells are of the same length at a given configuration. Considering the state of the grid at t^{j-1} is known, to solve the fluid problem, we have to compute at Runge-Kutta mid-step and micro time step, the cells volumes as well as the velocity at the cell centers and faces.

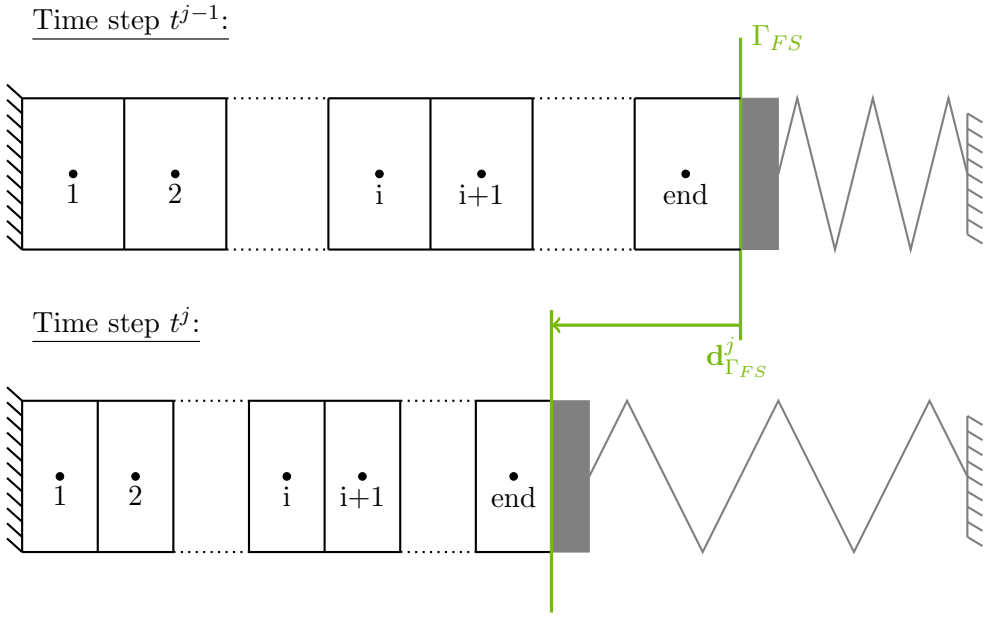


FIGURE 2.10: 1D piston problem grid deformation

First, we imposed that the velocity of the last face, called $end + \frac{1}{2}$, included in the fluid-structure interface Γ_{FS}, gets the same velocity as the fluid and the solid at this interface. Then, the velocity of the first face, numbered $\frac{1}{2}$, is imposed to zero. Then

a linear interpolation is performed to get the velocity of inner faces, such as:

$$\mathbf{w}_{i+\frac{1}{2}}^{j-\frac{1}{2}} = \frac{i}{end}\mathbf{v}_{\Gamma_{FS}}^{j-\frac{1}{2}} \tag{2.97a}$$

$$\mathbf{w}_{i+\frac{1}{2}}^{j} = \frac{i}{end}\mathbf{v}_{\Gamma_{FS}}^{j} \tag{2.97b}$$

Then, the velocities of cells centers are defined by the means of the velocities of the adjacent faces, $\mathbf{w}_i = \frac{1}{2}\mathbf{w}_{i+\frac{1}{2}} + \frac{1}{2}\mathbf{w}_{i-\frac{1}{2}}$.
Remark that here, in one dimension, velocities are all normal to the fluid structure interface, then $\mathbf{v}_{\Gamma_{FS}} = \mathbf{v}_s = \mathbf{v}_{f_{end+\frac{1}{2}}}$.
Finally, to determine the cell volumes, we have to update the position of the faces. As for velocities, the position of the first face is imposed to zero, the position of the last face is imposed as the position of the interface, and linear interpolation is made for the inner faces. The interface motion being defined as $\mathbf{d}_{\Gamma_{FS}}^{j} = \Delta t_f \mathbf{v}_{\Gamma_{FS}}^{j}$, we get:

$$x_{i+\frac{1}{2}}^{j-\frac{1}{2}} = x_{i+\frac{1}{2}}^{j-1} + \frac{1}{2}\Delta t_f \frac{i}{end}\mathbf{v}_{\Gamma_{FS}}^{j-\frac{1}{2}} \tag{2.98a}$$

$$x_{i+\frac{1}{2}}^{j} = x_{i+\frac{1}{2}}^{j-1} + \Delta t_f \frac{i}{end}\mathbf{v}_{\Gamma_{FS}}^{j} \tag{2.98b}$$

Thus, during the updating of the ALE grid, the volume of the cell i is computed as $\Delta V_i = a(x_{i+\frac{1}{2}} - x_{i-\frac{1}{2}})$, where a is the chamber section, which is 1 in this one dimensional test case.

2.3.2 Partitioned approach

In order to compare the results of the proposed method, the piston problem is also solved with a partitioned serial explicit coupling. The term explicit has the meaning here of direct, concerning the coupling. In opposition of implicit coupling where there are several iterations between the fluid and the solid solution at the same time step, see section 1.1.2. Also, the coupling is considered as serial due to the fact that the sub-domains are solved one after the other and none in a parallel way. The partitioned coupling method used here is the same as the one introduced in [41].
Obviously, an implicit coupling method will give better results in terms of accuracy. Nevertheless, the idea is here to compare two direct coupling methods, with only one computation per time step of each sub-domain. Moreover the comparison of the result will be done only for mono time scale FSI coupling.
The method of resolution for the piston problem by a partitioned serial explicit and mono time scale, is described below.

As said previously, partitioned approaches treat fluid and solid sub-domains in an uncoupled way, meaning that interactions between the sub-domains are implemented as boundary conditions. Thus, at the interface, the normal solid velocity is imposed to the fluid, and the fluid pressure is imposed to the solid. Then, the fluid variables need to be defined at the boundaries. As mentioned in section 2.1.2, the fluid variables are defined at faces of the grid as the mean of the variables at the center of the adjacent cells. Thus, in this one dimensional test case, fluid variables, as velocity or pressure

noted q_f, are approximated at the face $i + \frac{1}{2}$ as:

$$q_{f_{i+\frac{1}{2}}} = \frac{1}{2} q_{f_i} + \frac{1}{2} q_{f_{i+1}} \tag{2.99}$$

Nevertheless, for the first face, numbered $\frac{1}{2}$, and the last face, numbered $end + \frac{1}{2}$, there is only one adjacent cell, 1 and end respectively. This is why, fictitious cells are defined adjacent at the boundaries, outside of the fluid domain. There are called ghost cells and numbered 0 and $end + 1$, see Fig.2.11.

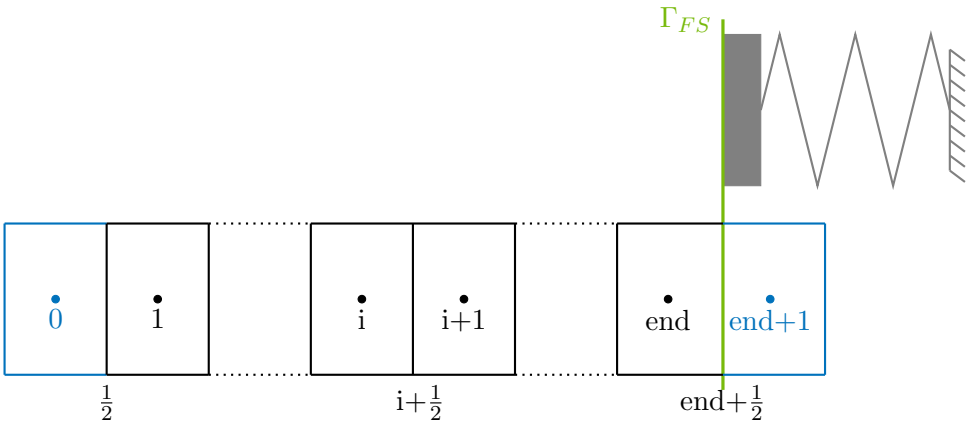


FIGURE 2.11: 1D piston problem ghost cells definition

Then, conservative variables are imposed at ghost cells as one order extrapolation from the inner cells and the boundary conditions.

The resolution of the piston problem is mono time scale. Thus, the objective is to compute the behaviour of fluid and structural sub-domains at each discrete instant in $[0, ..., t^n, t^{n+1}, ..., T]$ with $t^{n+1} = \Delta t + t^n$ and $\Delta t = 2 \times 10^{-5}\,\mathrm{m\,s^{-1}}$. Moreover the sub-domains are discretized using the numerical methods as for the proposed coupling method and described in section 2.1. After the initialization of the solid state vector and conservative fluid variables vector, using the initial condition and boundary conditions, the following procedure is computed for each time step.
First the ALE grid is updated at the Runge-Kutta mid-step $t^{n+\frac{1}{2}}$ and at the next t^{n+1}. To solve equations (2.97) and (2.98) in order to know the grid velocities and the cells volume, the interface velocity should be known at t^{n+1} and $t^{n+\frac{1}{2}}$, which is not yet the case. In this way, those values are predicted. The prediction of $\mathbf{v}_{\Gamma_{FS}}^{n+1}$ is defined as a one order extrapolation from the previous time steps. And the interface velocity at Runge-Kutta mid-step is considered as the linear interpolation between t^n and t^{n+1}, such as:

$${}^{p}\mathbf{v}_{\Gamma_{FS}}^{n+1} = 2\mathbf{v}_{\Gamma_{FS}}^{n} - \mathbf{v}_{\Gamma_{FS}}^{n-1} \tag{2.100a}$$

$${}^{p}\mathbf{v}_{\Gamma_{FS}}^{n+\frac{1}{2}} = \frac{3}{2}\mathbf{v}_{\Gamma_{FS}}^{n} - \frac{1}{2}\mathbf{v}_{\Gamma_{FS}}^{n-1} \tag{2.100b}$$

Now that the ALE has been updated, the fluid behaviour can be computed, using equation (2.65). Then, the conservative variables vector has to be defined at ghost cells centers, at $t^{n+\frac{1}{2}}$ and t^{n+1}. The ghost cell $end+1$ imposed the boundary condition of the solid sub-domain over the fluid. The fluid velocity at the boundary Γ_{FS} should be the solid velocity. As for the ALE mesh update, the velocity of the interface

computed by the solid is still unknown at the current time step. Then, instead of the computed solid velocity, the predictive velocity at the interface defined by equations (2.100), is used to impose the normal velocity condition trough the interface. The other conservative variables are extrapolated from inner cells. Then, the conservative variables vector at the right ghost cell is defined as:

$$\mathbf{U}_{f_{end+1}}^{n+\frac{1}{2}} = \begin{bmatrix} 2\rho_{f_{end}}^{n+\frac{1}{2}} - \rho_{f_{end-1}}^{n+\frac{1}{2}} \\ \rho_{f_{end+1}}^{n+\frac{1}{2}}(2^p\mathbf{v}_{\Gamma_{FS}}^{n+\frac{1}{2}} - \mathbf{v}_{f_{end}}^{n+\frac{1}{2}}) \\ 2E_{f_{end}}^{n+\frac{1}{2}} - E_{f_{end-1}}^{n+\frac{1}{2}} \end{bmatrix} \qquad \mathbf{U}_{f_{end+1}}^{n+1} = \begin{bmatrix} 2\rho_{f_{end}}^{n+1} - \rho_{f_{end-1}}^{n+1} \\ \rho_{f_{end+1}}^{n+1}(2^p\mathbf{v}_{\Gamma_{FS}}^{n+1} - \mathbf{v}_{f_{end}}^{n+1}) \\ 2E_{f_{end}}^{n+1} - E_{f_{end-1}}^{n+1} \end{bmatrix}$$

(2.101) (2.102)

We also have to take into account the wall boundary condition and define the conservative variables vector at the left ghost cell center. The objective is to enforce a zero velocity at the fixed wall,$v_{f_{\frac{1}{2}}} = 0$. Then we imposed:

$$\mathbf{U}_{f_0}^{n+\frac{1}{2}} = \begin{bmatrix} 2\rho_{f_1}^{n+\frac{1}{2}} - \rho_{f_2}^{n+\frac{1}{2}} \\ -\rho_{f_0}^{n+\frac{1}{2}}\mathbf{v}_{f_1}^{n+\frac{1}{2}} \\ 2E_{f_1}^{n+\frac{1}{2}} - E_{f_2}^{n+\frac{1}{2}} \end{bmatrix} \quad (2.103) \qquad \mathbf{U}_{f_0}^{n+1} = \begin{bmatrix} 2\rho_{f_1}^{n+1} - \rho_{f_2}^{n+1} \\ -\rho_{f_0}^{n+1}\mathbf{v}_{f_1}^{n+1} \\ 2E_{f_1}^{n+1} - E_{f_2}^{n+1} \end{bmatrix} \quad (2.104)$$

In a second phase, the solid state is computed at t^{n+1}. Equations (2.31) are solved where the mass and stiffness matrices are replaced by the scalar values m_s and k_s respectively in this one dimensional problem. Also, an external force is applied which is the interaction of the fluid onto the solid at the interface and is defined as:

$$\mathbf{F}_s^{n+1} = a(p_{f_{end+\frac{1}{2}}}^{n+1} - p_f^0) \quad (2.105)$$

Where the fluid pressure at the interface $p_{f_{end+\frac{1}{2}}}^{n+1}$ is defined thanks to the ghost cell. The algorithm 3 describes the serial explicit partitioned coupling method.

2.3.3 Monolithic co-simulation method

On the other side, the piston problem is computed by the monolithic co-simulation method presented in section 2.2. The method is not described once again, applied to the piston problem, due to the fact that it follows the algorithm 2. Nevertheless, here is some precision.

First, concerning the solid resolution, the equations to solve are the same as (2.31), except that, due to the one dimensional assumption, the mass matrix $\mathbf{M}_s$ and stiffness matrix $\mathbf{K}_s$ are replaced by the scalar mass m_s and stiffness k_s of the spring.

Then, the ALE grid update is the same as the one used for the partitioned coupling, solved by the equation (2.97) and (2.98). Where the interface velocity is not predicted but computed by the equations (2.92 2.93 2.87) and the ghost cell interpolation, due to the fact that in one dimension every velocity is normal to the interface, such as $\mathbf{V}_{\Gamma_{FS}} = \mathbf{V}_s = \mathbf{V}_{f_{end+\frac{1}{2}}}$.

Algorithm 3 Partitioned serial explicit coupling

Initial fluid Ω_f and solid Ω_s states
while $t^{n+1} \leq T$ **do** ▷ Temporal loop
  Extrapolate ${}^p\mathbf{v}_{\Gamma_{FS}}^{n+\frac{1}{2}} \leftarrow$ (2.100a) ▷ Runge-Kutta first step
  Compute $\Delta V^{n+\frac{1}{2}}, \mathbf{w}^{n+\frac{1}{2}} \leftarrow$ (2.97a 2.98a)
  Compute $\mathbf{U}_f^{n+\frac{1}{2}} \leftarrow$ (2.66a)
  Extrapolate $\mathbf{U}_{f_{end+\frac{1}{2}}}^{n+\frac{1}{2}} \leftarrow$ (2.101)
  Extrapolate $\mathbf{U}_{f_{\frac{1}{2}}}^{n+\frac{1}{2}} \leftarrow$ (2.103)
  Extrapolate ${}^p\mathbf{v}_{\Gamma_{FS}}^{n+1} \leftarrow$ (2.100a) ▷ Runge-Kutta second step
  Compute $\Delta V^{n+1}, \mathbf{w}^{n+1} \leftarrow$ (2.97a 2.98a)
  Compute $\mathbf{U}_f^{n+1} \leftarrow$ (2.66a)
  Extrapolate $\mathbf{U}_{f_{end+\frac{1}{2}}}^{n+1} \leftarrow$ (2.101)
  Extrapolate $\mathbf{U}_{f_{\frac{1}{2}}}^{n+1} \leftarrow$ (2.103)
  compute $\mathbf{p}_{f_{end+\frac{1}{2}}}^{n+1} \leftarrow$ (2.105)
  compute $\mathbf{U}_s^{n+1} \leftarrow$ (2.37)
end while

Finally, the coupling condition are not imposed thanks to the ghost cells method, but conservative variables vector still has to be defined at ghost cells to compute the numerical flux at the boundaries and interpolate fluid velocities at the interface. Thus, all the values of the conservative variables vector are extrapolated for the right ghost cell such as:

$$\mathbf{U}_{f_{end}}^{n+\frac{1}{2}} = 2\mathbf{U}_{f_{end-1}}^{n+\frac{1}{2}} - \mathbf{U}_{f_{end-2}}^{n+\frac{1}{2}} \tag{2.106a}$$

$$\mathbf{U}_{f_{end}}^{n+1} = 2\mathbf{U}_{f_{end-1}}^{n+1} - \mathbf{U}_{f_{end-2}}^{n+1} \tag{2.106b}$$

Concerning the left ghost cell, this one is defined as in the partitioned method, equations (2.103, 2.104), in order to impose the fixed wall boundary condition.

2.3.4 Energy balance

Those two coupling methods are implemented into python scripts and the results are described in the next sub-section. Moreover, the energy balance of the coupling problem will be studied and is presented below.
The mechanical energy $\mathscr{E}$ over each sub domain is written as:

$$\mathscr{E}_k(t) = \mathscr{W}_{kin_k}(t) + \mathscr{W}_{int_k}(t) - \mathscr{W}_{ext_k}(t) \qquad \forall t \in [0,T] \qquad k = s, f \tag{2.107}$$

Where $\mathscr{W}_{kin}$ is the kinetic energy, $\mathscr{W}_{int}$ the internal energy and $\mathscr{W}_{ext}$ the external energy. In the considered problem the external energy over each sub-domain in only induced by the force of interface, no other external forces being considered.
First we considered the solid sub domain energy. Its internal and kinetic energies are

computed at the end of each micro time step such as:

$$W^m_{kin_s} = W^0_{kin_s} + \frac{1}{2}\mathbf{v}^m_s m_s \mathbf{v}^m_s - \frac{1}{2}\mathbf{v}^0_s m_s \mathbf{v}^0_s \tag{2.108}$$

$$W^m_{int_s} = W^0_{int_s} + \frac{1}{2}\mathbf{d}^m_s k_s \mathbf{d}^m_s - \frac{1}{2}\mathbf{d}^0_s k_s \mathbf{d}^0_s \tag{2.109}$$

Concerning the fluid sub domain, its total energy is computed at the end of every thinner time step, not at the Runge-Kutta mid-step. It is defined by the equation (2.40) and is computed by the resolution of the conservative Euler equation. Thus, the total energy over the fluid sub-domain W_f is resolved as:

$$W^j_f = W^{j-1}_f + \sum_{i=1}^{n_{cell}} \Delta V^j_{f_i} E^j_{f_i} - \sum_{i=1}^{n_{cell}} \Delta V^{j-1}_{f_i} E^{j-1}_{f_i} \tag{2.110}$$

Finally, the external coupling is computed differently according to the coupling method used.
During the serial explicit coupling, the external energy is expressed according to the interface force $\mathbf{F}^{n+1}_s$ from equation (2.105), such as:

$$W^{n+1}_{ext} = W^n_{ext} + \frac{1}{2}(\mathbf{F}^{n+1}_s + \mathbf{F}^n_s)(\mathbf{d}^{n+1}_s - \mathbf{d}^n_s) \tag{2.111}$$

In case of monolithic co-simulation coupling, the external energy is computed by the mean of the Lagrange multipliers, and are called interface energy $W_{\Gamma_{FS}}$. For the solid sub-domain balance, it is defined at the macro time step as:

$$W^m_{\Gamma_{FS_s}} = W^0_{\Gamma_{FS_s}} + \frac{1}{2}(\mathbf{\Lambda}^m + \mathbf{\Lambda}^0)(\mathbf{d}^m_s - \mathbf{d}^0_s) \tag{2.112}$$

For the fluid sub-domain balance, the contribution of the interface is computed every micro time step. Instead of the solid displacement, the last face of the ALE grid, which is included into the fluid-structure interface and whose position is computed at the micro time scale, is used.

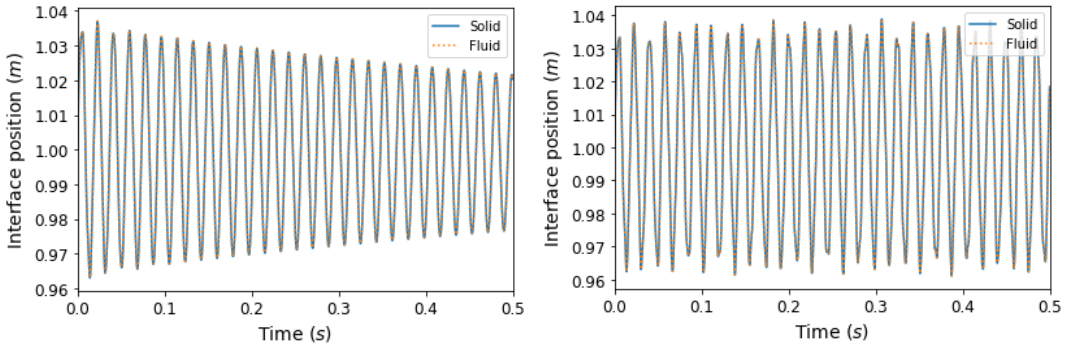
$$W^j_{\Gamma_{FS_f}} = W^{j-1}_{\Gamma_{FS_f}} + \frac{1}{2}(\mathbf{\Lambda}^j + \mathbf{\Lambda}^{j-1})(\mathbf{x}^j_{\Gamma_{FS}} - \mathbf{x}^{j-1}_{\Gamma_{FS}}) \tag{2.113}$$

Finally, the energy balance of each sub-domain is known by equation (2.107). Since no external energy is added to the system, the sum of the fluid and solid balance should remain constant.

2.3.5 Results

First we compare the results of the computation of the piston problem by the proposed coupling method, a monolithic co-simulation approach, and by the serial explicit partitioned coupling, with the same time scale for the fluid sub-domain and for the solid sub-domain.
Fig.2.12 shows the position of the interface Γ_{FS} according to the time. The graph on the left is the result from the partitioned coupling simulation while the one on the right comes from the computation of the proposed coupling method. First, the

positions of the interface computed with the solid displacement, the blue line, and by the mean of the ALE grid updating, the dashed orange line, seem to be really close for both methods, which validates the implementation of coupling condition and ALE grid computation. To validate the sub-domains computation, the general appearance of the results, maximum amplitudes and frequencies, are compared to the results from the literature. As said earlier, in order to compare our results, the set of parameters used are the sames as the one used by Blom in [16] and by Ischinger in [59], whose results are shown in Fig.2.13. The behaviour of our test case piston computed by both methods matches with the literature. Thus, the computational methods of each sub-domain and the coupling appears to be validated. The main difference of the interface position evolution concerns the amplitude. In Fig.2.12a, the computation by the partitioned approach shows a large decrease of the amplitude while the amplitude seems to be stable in Fig.2.12b computed by the proposed method. Let's recall that the spring is undamped, and the fluid is inviscid. Thus, the interface amplitude should not decrease over time since there is no dissipative work in the system. In this way, the amplitude decrease in the partitioned coupling doesn't come from a loss of physical energy but from numerical energy, induced by the coupling method.
A certain amount of amplitude loss is also observed on the results from literature, even though it is very small with the method proposed by Ischinger which used Discontinuous Galerkin method, as showed on Fig.2.13b. Then the proposed method of an FSI coupling based on monolithic formulation seems to reach the objective of being conservative, contrary to the serial explicit partitioned coupling.



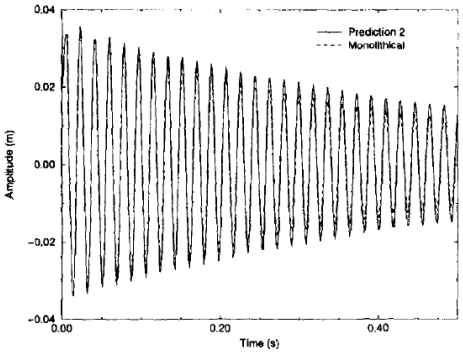
(A) Serial explicit coupling (B) Monolithic co-simulation coupling

FIGURE 2.12: Position of the interface according the time computed by solid (blue line) and fluid (dashed orange line) sub-domains

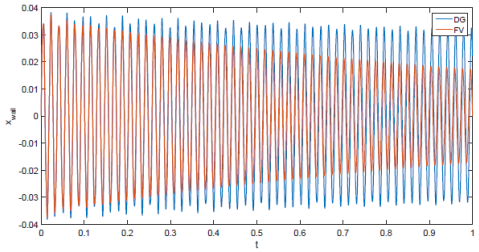
Then we study the relative error of coupling for the velocities at the interface. This one is computed as:

$$err = \frac{|\mathbf{v}_s^{n+1} - \mathbf{v}_{f_{\Gamma_{FS}}}^{n+1}|}{\max(\mathbf{v}_s)} \tag{2.114}$$

Fig.2.14 below shows the evolution of this values according to time. Fig.2.14b shows that the relative coupling error is of computer's accuracy order. This means that the continuity of the velocities through the fluid-structure interface seems to be well implemented. Concerning the partitioned coupling, the relative error is average of 2.5%, which is not that bad but not as accurate as the proposed coupling method, while both methods impose the velocity continuity. Indeed, the serial explicit partitioned coupling imposes that the fluid velocity at the interface be equal to the solid velocity, by the means of the ghost cell method, see (2.101) and (2.102). Nevertheless, due



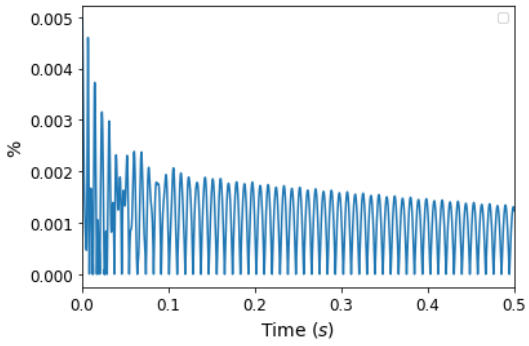
(A) Amplitude from 0s to 0.5s



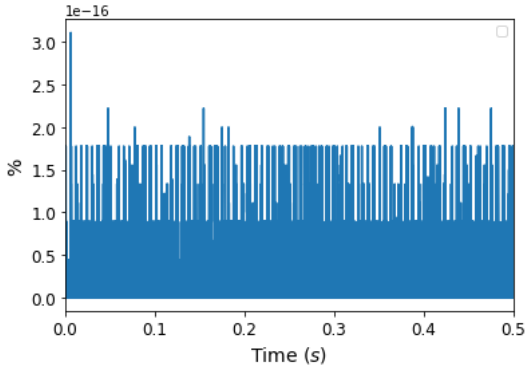
(B) Interface position from 0s to 1s

FIGURE 2.13: Some results from literature, on the left the interface amplitude [16] and on the right the interface position [59]

to the unknown solid velocity at this point of the algorithm, a prediction about it is used, explaining the relative coupling error.



(A) Serial explicit coupling

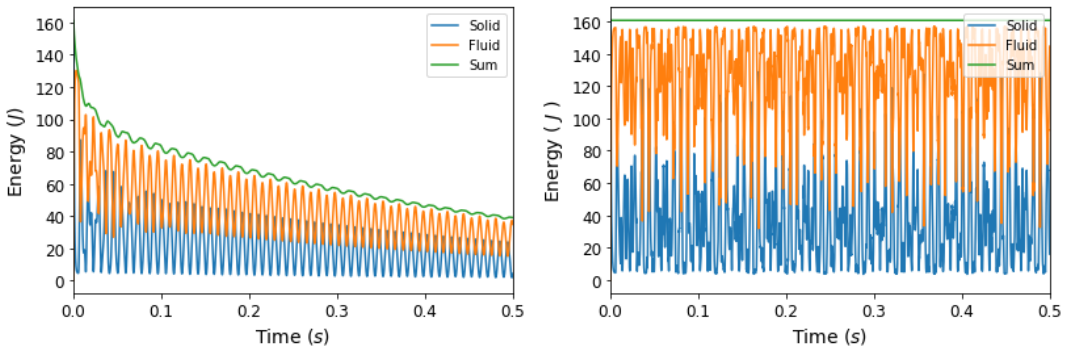


(B) Monolithic co-simulation coupling

FIGURE 2.14: Relative coupling error between fluid and solid velocities at the interface

The last data compared for the two coupling approaches are the energy balance. As said previously, energy preservation is one of the main issues of FSI simulation and the proposed method with mono time scale, should be conservative. Fig.2.15 shows the energy balance of both computations. The blue line is the solid energy, the orange line is the fluid energy, and the green one is the summation of the energy over the two sub-domains. In Fig.2.15b the total energy is constant according to time, which means that the proposed method is well conservative. While the energy balance of the partitioned coupling is decreasing, see Fig.2.15a. This validates the amplitude loss of the interface position coming from dissipative energy induced by the partitioned coupling method.

After the comparison of the simulation run by the partitioned serial explicit algorithm and the monolithic co-simulation algorithm with mono-time scales, let's study the results of the proposed method for multi-time scales simulation.
As said previously the thinner time scale is used to compute the fluid sub domain, and we keep $\Delta t_f = 2 \times 10^{-5}\,\mathrm{m\,s^{-1}}$. Then, the solid sub-domain is solved using different



(A) Serial explicit coupling (B) Monolithic co-simulation coupling

FIGURE 2.15: Energy balance over the fluid and structural sub-domains

time steps such as $\Delta t_s = m\Delta t_f$. The Fig.2.16 shows the position of the fluid-structure interface Γ_{FS} for several time step ratio m.

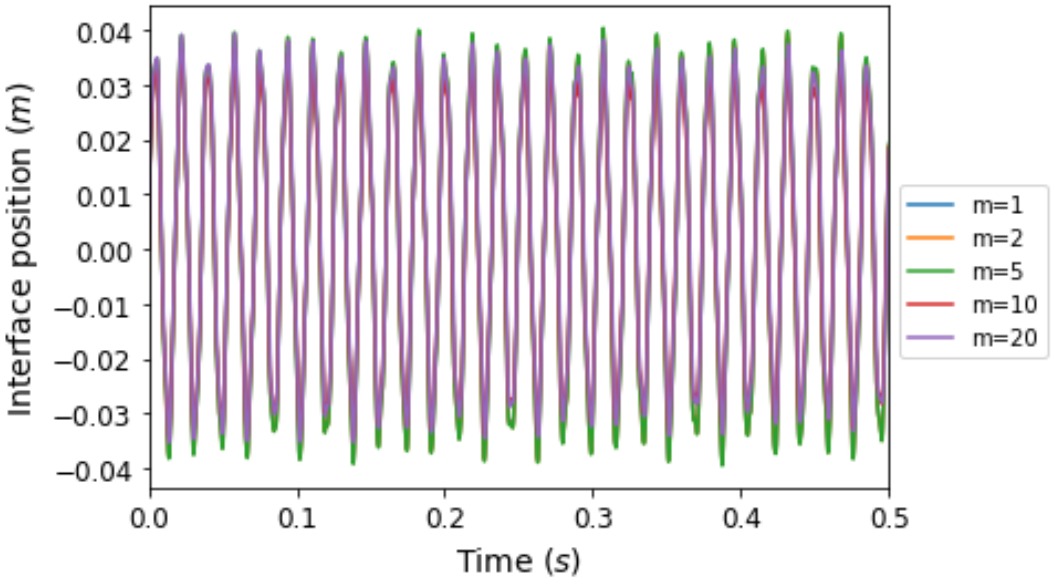
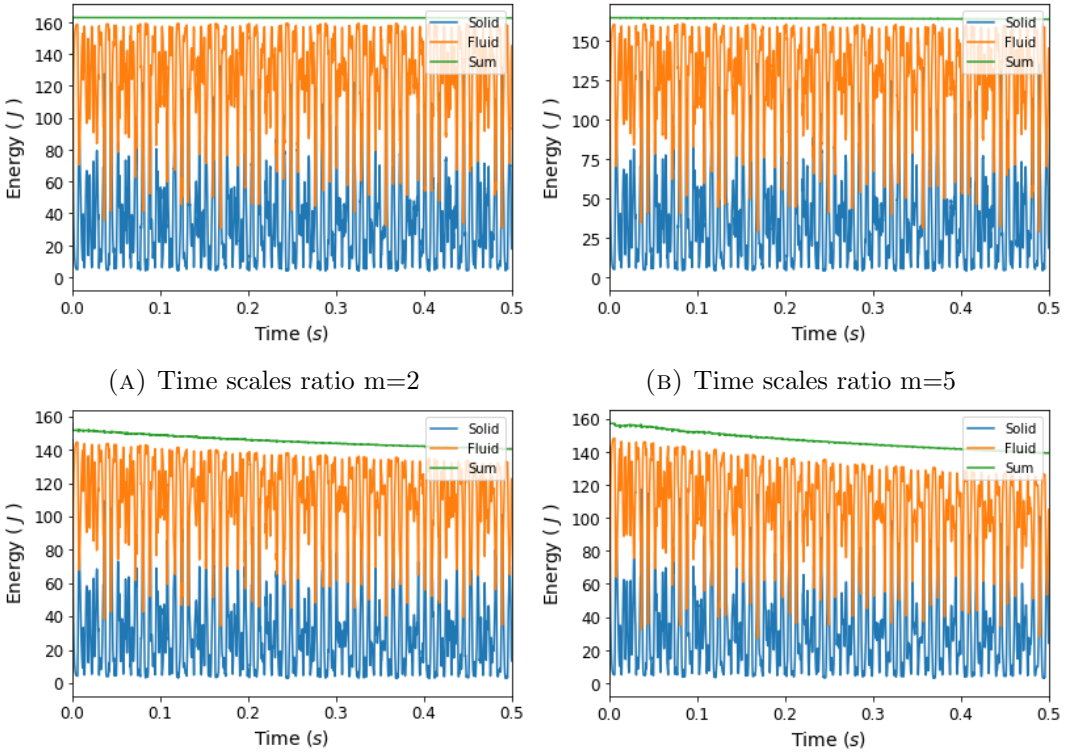


FIGURE 2.16: Displacement of the interface for different time scale ratios

We can see that the positions over time of the interface for $m = 1, 2, 5, 10, 20$ are quite similar. Nevertheless, the results seem to deteriorate with the increasing of m. In particular since $m = 10$, where we can see the amplitude decrease significantly. As said in section 2.2, we chose to impose the continuity of the velocities at the micro time scale that can induce a small numerical dissipation energy, while ensuring the global stability.
Indeed, Fig.2.17 shows energy balance of the piston simulation with different time scales, and we can see the global energy decreasing over time. For $m = 2$ and $m = 5$, the loss of energy can be considered negligible. This one is really larger for $m = 10$ and $m = 20$, and can not be neglected. Nevertheless, regarding the time ratio, it is not that large, indeed for a factor 10 between the two time scales, the error of global energy is less than 5% at the end of the simulation.



(A) Time scales ratio m=2

(B) Time scales ratio m=5

(C) Time scales ratio m=10

(D) Time scales ratio m=20

FIGURE 2.17: Energy balance over the fluid and solid sub-domains

Conclusion

In this section, the proposed method has been tested thanks to the academic test case of the one dimensional piston. First results have been validated according to literature. Then performances of the proposed method with mono time scale have been compared with performances of partitioned serial explicit method. The proposed method is really more accurate since there is no deterioration of the amplitude according to time, contrary to the partitioned approach. Moreover, the method in mono time scale is conservative, the energy balance and the global stability of the coupling are thus guaranteed. Concerning the multi-time step coupling, with a small ratio between the two time steps, the results are really promising; global stability is ensured, and small dissipative energy is obtained. For large time steps ratio, such as $m = 10$, the degradation of the results are more important but could be interesting in some cases regarding the saved computation time.
After the validation of the proposed method over an academical test case, the method has to be applied to more complex and realistic simulations, computed with dedicated solvers.

Chapter 3

Implementation and numerical results

In this chapter, the implementation of the proposed FSI coupling method is presented, as well as numerical results on 3D test cases computations.
In the first place, the objective was to implement the coupling method into the software suite from Ansys, the industrial partner of the thesis. Ansys solutions offer three means to solve FSI problems, all based on partitioned formulation. The first developed was the system coupling tools [23]. This one works as a black box coupling; the fluid finite volume solver Fluent [7] is coupled to the solid finite element solver Mechanical [6] through the coupling box. This one handles data exchange for partitioned one-way, two-way explicit and two-way implicit coupling, with an interface tracking approach. The coupling box also manages incompatible meshes at the fluid-structure interface.
The second method is an intrinsic FSI solver from the Fluent software. A simple solid linear elastic solver with first-order finite elements, has been developed inside the Fluent environment. Then, a two way serial explicit coupling with compatible meshes can be run.
The last method is fluid-structure solutions from LS-DYNA solver [98], acquired by Ansys Inc in 2019. Several methods are available to solve FSI problems, which couples with a partitioned formulation, a solid explicit finite element solver with different fluid solvers, in LS-DYNA environment. The first one is an incompressible flow solver (ICFD) [31], which is based on finite element method for incompressible fluid. The coupling between the two sub-domains is based on interface tracking and is either explicit or implicit. For compressible fluid, Conservative Element/Solution Element method (CESE) and dual-CESE solvers, satisfying the fluid conservative law, can be used [121]. Interface tracking approach or interface capturing approach, thanks to the immersed boundary method, can be used for the FSI coupling. The dual-CESE solver uses a dual mesh approach that allows to achieve a better stability than the CESE solver for the same element size.

First, the coupling box solution has been considered. As an existing complete code coupling solution, it seemed suitable for the implementation of the MCS method. Nevertheless, this tool is complex, and the access to the code is limited. In this regard, the use of the system coupling tool was not appropriate for a first implementation and has been given up. In order to use a simpler environment, the intrinsic Fluent solver solution has been considered in a second time. The issue of this approach is that it cannot be easily extended to more complex FSI problems, such as non-linear solid, incompatible meshes at the interface or multi-time scale approach. That is

why this solution has also been left aside. Finally, the LS-DYNA solutions for FSI problems have not been studied due to the fact that the acquisition occurred during the thesis. Therefore, a direct coupling between Fluent and Mechanical, developed from scratch to implement the MCS method seemed to be the more appropriate solution. The fields would be exchanged by writing and reading function added to each solvers, User Define Function (UDF) for Fluent and Ansys Parametric Design Language (APDL) for Mechanical. Moreover, the interface operator computation would be done into UDF. After some tries, this solution was hard to set up without direct access to solvers codes. Besides, the field exchange would not have been sufficiently efficient and the FSI problem size would have been limited. In addition, the extension to FSI problems with multi-time and space scale would have been complex.

Thus, even if the objective remains to implement the proposed co-simulation algorithm to existing fluid and solid solvers as little intrusively as possible, the access to the solvers codes is still required. Hence, we chose to couple existing open-source solvers but with the constraint that they were based on the same discretization methods as Fluent and Mechanical. Finally, instead of a direct coupling between physical solvers, a coupling environment was adopted in order to robustly manage the field exchanges, incompatible meshes and different time scales. The coupling Precise Code Interaction Coupling Environment (preCICE) was chosen [20].
First, this chapter presents the preCICE coupling library and the used solid and fluid solvers. Then the implementation of the proposed coupling method is detailed. Finally, numerical results corresponding to two benchmarks of the forward step and the perpendicular flap, are shown.

3.1 Presentation of coupling library and used solvers

The aim of this section is to present how preCICE library works for FSI simulations and its existing features. Then, the chosen free software used for sub-domains simulation are presented; their solvers and discretization methods, as well as their adapters, employed by preCICE for performing FSI simulation.

3.1.1 The coupling library preCICE

The first implementation of preCICE library dates from 2014, with the main contribution carried out in Gatzhammer's PhD thesis [47], based on an extension of the software "FSI*". In 2016, Ukerman proposed in his PhD thesis [114] major improvements about communication and architecture of the library, as well as parallelization of the coupling. Since that, the library mainly has kept the same framework and is maintained and developed at the University of Stuttgart and the Technical University of Munich.
Figure Fig.3.1 from preCICE documentation, presents an overview of the utilities and functioning of the library. The library proposes to couple physical solvers with a peer to peer approach. The solvers communicate directly with each other, by the mean of coupling adapters based on application programming interface (API). The API is natively in C++, but bindings for C, FORTRAN, Matlab and Python exist. The library can perform several multi-physics computations, but is only used in this work for FSI problems composed of two sub-domains. The library preCICE steers the coordination of the coupled simulation and data transfer. Several partitioned

coupling schemes are available. Furthermore, the meshes at the fluid-solid interface can be incompatible. The preCICE features are described succinctly below.

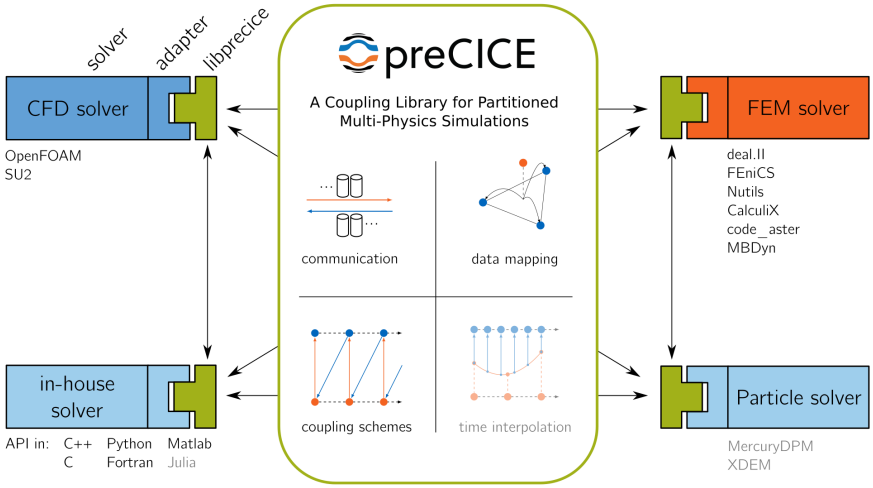
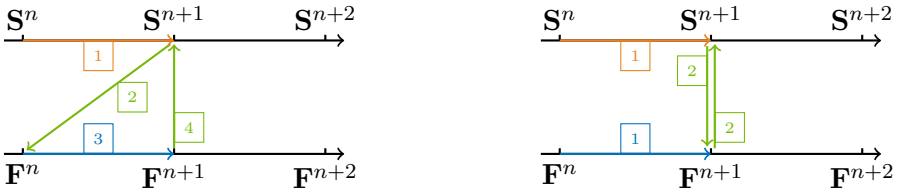


FIGURE 3.1: Overview of the library preCICE features

Coupling schemes

The preCICE library allows to run FSI simulation based on a partitioned formulation, see section 1.2.2. Concerning the explicit schemes, they can be serial, based on explicit conventional serial staggered (CSS) algorithm, or parallel, using the explicit conventional parallel staggered (CPS) algorithm, see Fig.3.2.



(A) CSS (B) CPS

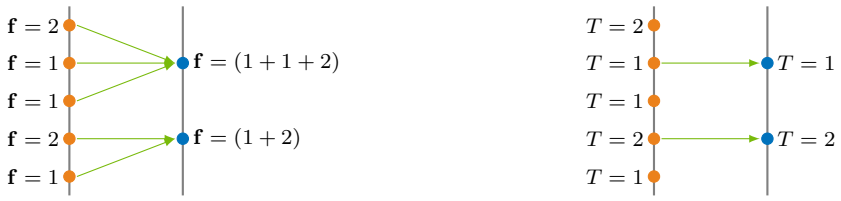
FIGURE 3.2: Partitioned explicit schemes available in preCICE

Implicit coupling schemes, serial or parallel, are also available into the preCICE library. Each solver is run until the defined convergence criterion is satisfied or when the maximum number of coupling iterations is reached. The implicit coupling can use SIMPLE under relaxation method, adaptive Aitken method or several Quasi-Newton methods.

Data mapping

The preCICE library allows the coupling between two incompatible meshes at the interface; the mesh surfaces fit, based on interface tracking method, but the nodes on either side are not shared.
The mapping between the two meshes has to verify one of these constraints, consistent or conservative mapping. With a conservative mapping, the value at a node belonging to the discrete coarse surface is computed as an aggregation of the corresponding nodes belonging to the fine discrete surface, see Fig.3.3a. This constraint is used for

quantities that are absolute, such as force or mass quantities. On the other hand, with one consistent mapping, the value at a coarse node is the same as the value at the corresponding fine nodes, see Fig.3.3 b. This constraint is imposed to normalize quantities such as temperature or pressure.



(A) Conservative mapping, for forces f, as instance

(B) Consistent mapping, for temperature T, as instance

FIGURE 3.3: Data mapping constraint on nearest-neighbor method

Three mapping methods can be used with preCICE. The easiest and fastest mapping method is the nearest-neighbor method which is first order. The nearest-projection method, is a bit more complex, due to the fact that nodal connectivity is required. Nevertheless, it is mostly order two. This method uses linear interpolation within each element, see Fig.3.4b Finally, Radial-Basis Function mapping has been implemented into the library thanks to Lindner's PhD thesis, more detailed can be found in [74].

(A) Nearest neighbor

(B) Nearest projection

FIGURE 3.4: Data mapping methods

The mapping can be computed only once at the initialization stage, which can be sufficient for small interface deformation. In case of large deformation, the mapping can be re-computed at each time step or simply on demand, to reduce the computation cost.

Time interpolation

The time interpolation feature is still under development. In the case of different time steps adopted by the fluid and the solid solvers, preCICE can manage the proceeding of the coupling simulation. However, variables exchanged are not interpolated at the finer time step; they are considered fixed. In this case, the exchanged fields from the solver using the larger time scale to the solver using the finer timer scale are not updated for the micro time step computation.

Communication

Two protocols can be used for the communication between the two solvers called also participants. The communication channel is based on TCP/IP sockets by default,

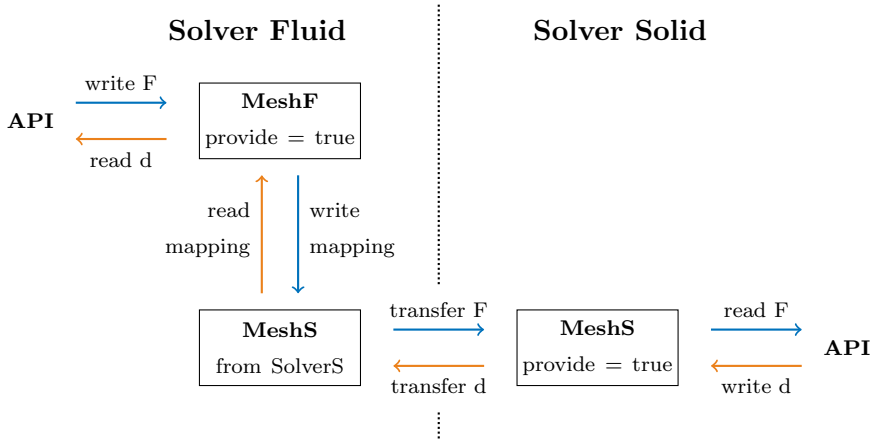


FIGURE 3.5: Overview of data exchange procedure for FSI coupling with preCICE

but MPI ports can also be used, even if not recommended.
Let us describe more precisely the data exchanges between the two participants for a partitioned FSI simulation using preCICE, as shown in Fig.3.5. The fluid solver uses solid displacement of the interface as Dirichlet boundary conditions, whereas the solid solver uses the fluid forces onto the interface as Neumann boundary conditions. Each participant uses its own mesh for which its physical solution is computed. Then, to allow data exchanges, one of the participants also has to use the mesh from the other participant. In this example, the fluid solver uses the fluid mesh `MeshF` and the solid mesh `MeshS`, while the solid solver only uses `MeshS`. Thus, data are mapped between the two meshes of the fluid participant, `MeshF` provided by the fluid solver and `MeshS` from the solid participant. Then, data are transferred between the two participants on `MeshS` which is shared by both participants.

Thus, the preCICE library allows non-intrusive coupling of Fluid and Solid solvers, for partitioned coupling with non-matching meshes. The configuration of the coupling is written into an `XML` file which is read during the initialization stage of the coupling. An example of `XML` file used for the coupling configuration in the simulations presented in the following can be found in Appendix C.1.

3.1.2 Solid solver: CalculiX

The preCICE library allows for the use of several solid dynamic solvers for multi-physics simulation. We chose to use CalculiX; the next subsection describes this software and then the adapter developed to run it with preCICE.

CalculiX is a free software developed in C and Fortran by Guido Dhont [33]. Its naming conventions and input style formats are based on the software Abaqus. CalculiX proposes solvers based on FEM for continuum mechanics simulation. Mechanical, thermal fluid and electromagnetic solvers are available. The simulation run can be static, modal or dynamic. The software is divided into two modules. CalculiX GraphiX (cgx) has pre-processing capabilities and is used for post-processing. CalculiX CrunchiX (ccx) manages pre-processing and solution.
In this work, CalculiX is used as the solid dynamic solver only, in order to calculate

the response of a structure subjected to dynamic loading using a direct time integration of the equations of motion. Concerning the spatial discretization, more than fifty types of finite elements are available. In this thesis, regarding the linear elasticity assumption for the solid sub-domain, and the 3D problem explored in this work, only C3D8 elements are used. The C3D8 element is a general purpose linear brick element, fully integrated. Concerning the time integration scheme, GalculiX uses the generalized α-method [25], where the α parameter is such as $\alpha \in [-\frac{1}{3}, 0]$. Choosing $\alpha = 0$, the implicit Newmark scheme is retrieved, as used in section 2.1.1.

As said in the introduction of this section, the preCICE library uses a peer to peer approach; the solvers are communicating directly with each other, through the API of the coupling adapters. That is why the use of CalculiX for FSI coupling in the preCIE environment requires an adapter. There are three ways for preCICE adapters implementation: direct modification (not recommended), adapter class and callback functions.
The adapter for CalculiX is based on the adapter class method. The implementation has been proposed first by Cheung [22] for thermal coupling and has been extended to FSI coupling later [113]. A new routine `nonlingeo_precice.c` is created and called instead of `nonlingeo.c`, which is called for nonlinear static or dynamic calculation, for preCICE simulations. `nonlingeo_precice.c` is identical to `nonlingeo.c` except that some preCICE functions are added. In the general case, any adapter has at least to call three preCICE functions: `initialize` which establishes the communication channels and sets up the coupling configuration, `advance`, which is called after every time step to advance the coupling and `finalize` which frees the preCICE data solid and closes the communication channels. The pseudo algorithm of the CalculiX adapter is summarized in Alg.4.

Algorithm 4 CalculiX adapter

```
1: ccx initialization
2: Precice_Setup                                   ▷ initialize is called
3: while Precice_IsCouplingOngoing do              ▷ Temporal loop
4:     Precice_AdjustSolverTimestep
5:     Precice_ReadCouplingData
6:     if Precice_IsWriteCheckpointRequired then   ▷ for implicit coupling
7:         Save checkpoint
8:     end if
9:     ccx solid computation
10:    Precice_WriteCouplingData
11:    Precice_Advance
12:    if Precice_IsReadCheckpointRequired then    ▷ coupling not convergence
13:        Load variables from checkpoint
14:    else
15:        end time step
16:    end if
17: end while
18: PreCICE_FreeData                               ▷ finalize is called
```

In this way, CalculiX uses the same time discretization methods as used for the demonstrator of the MCS method in the second chapter and can be employed into the preCICE environment thanks to its adapter.

3.1.3 Fluid solver: OpenFOAM

Concerning the fluid software, available adapters to preCICE are less numerous as for solid solvers. An adapter exists for the Open-source Field Operation and Manipulation (OpenFOAM) software [117], which will be used in this work.

OpenFOAM is a C++ free software which uses a high level programming and a friendly syntax for partial differential equations. It proposes solvers for continuum mechanics problems (most prominently including CFD), based on FVM.
More than seventy solvers are distributed. Concerning CFD solvers, they can be divided into two categories adopting distinct approaches. Most of CFD solver, from OpenFOAM and other software, are pressure based solvers. This approach has been the first developed and is historically used to compute incompressible fluid with low Mach number. This kind of solvers computes the fluid variables with a sequential approach, where the pressure and the velocity are coupled in an elliptic manner. The flowchart of this kind of algorithm is shown in Fig.3.6.

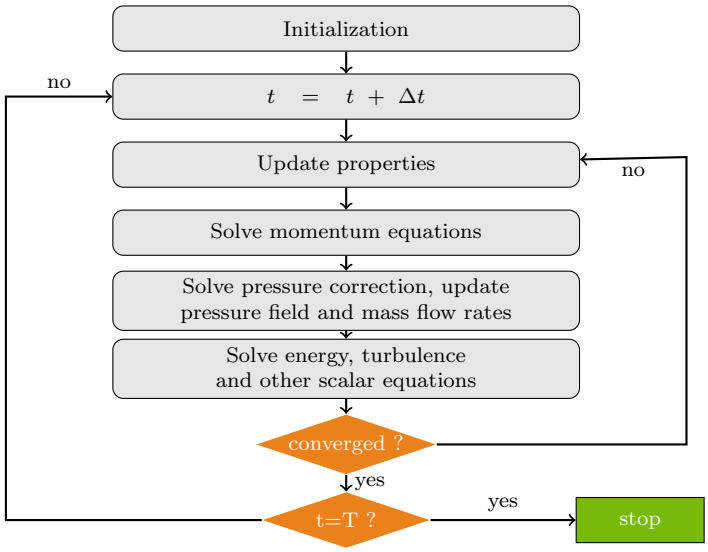


FIGURE 3.6: Pressure based CFD solvers flow chart

The other approach developed later is called density based. This type of solvers is mainly used for compressible flow with high Mach number. The density based solvers use a global approach where the conservative variables that are density, momentum and volumic energy are solved simultaneously, see Fig.3.7. Regarding the proposed MCS coupling, which is based on energetic methods, the second kind of solvers, density based ones, seems to be really more appropriate. Moreover, using an explicit time integration scheme, no convergence loop is needed, as presented in 2.1.2, with the Runge-Kutta scheme of second order. Nevertheless, there is no solver from openFOAM which is density based, explicit and second order accurate. This is why we focus on the solver rhoCentralDymFaom, presented below.

The rhoCentralFoam solver has been proposed in [54]. It is a density based solver for transient inviscid or viscous, compressible flow. The space is discretized according to the cell centered finite volume method, and the numerical flux are computed by central-upwind schemes of Kurganov and Tadmor [67]. For the time discretization,

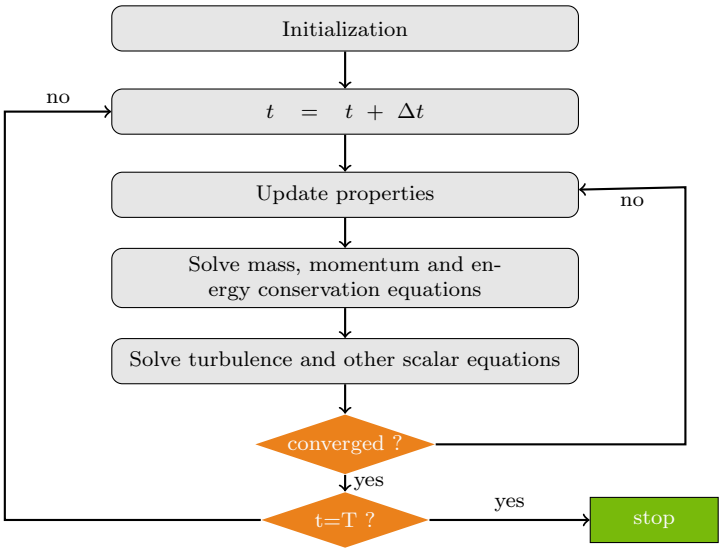


FIGURE 3.7: Density based CFD solvers flow chart

an explicit backward scheme is used, which can be assimilated as a one step Runge-Kutta scheme, of the order one.
The rhoCentralDymFoam is the ALE version of the rhoCentralFoam solver which is based on the Eulerian formulation. Numerous solutions can be used for the grid resolution, which is mobile and arbitrary deformed according to the motion of the Lagrangian moving wall, which plays the role of the interface in FSI problems.
Some simple modifications of the proposed coupling method in chapter 2, have to be done to fit the backward Euler scheme instead of the Runge-Kutta scheme. First, in the discretized equations of the global coupled problem (2.84), the fluid conservative equations are only written at t^j as:

$$\Delta V_{f_i}^j \mathbf{U}_{f_i}^j = \Delta V_{f_i}^{j-1} \mathbf{U}_{f_i}^{j-1} - \Delta t_f \sum \left(s_{f_{ih}}^{j-1} \tilde{\mathbf{F}}_{f_{ih}}^{j-1} + \mathbf{L}_{f_{ih}}^T \Lambda_{ih}^j \right) \tag{3.1}$$

Concerning the ALE grid updating, following the explicit first order accurate computation over the fluid sub-domain, the interface displacement can be written as:

$$\mathbf{d}_{\Gamma_{FS}}^j = \mathbf{d}_{\Gamma_{FS}}^{j-1} + \Delta t_f \mathbf{w}_{\Gamma_{FS}}^{j-1} \tag{3.2}$$

Then, as the interface velocity is enforced as the solid velocity at the interface, equation (3.2) becomes:

$$\mathbf{d}_{\Gamma_{FS}}^j = \mathbf{d}_{\Gamma_{FS}}^{j-1} + \Delta t_f \mathbf{v}_{s_{\Gamma_{FS}}}^{j-1} \tag{3.3}$$

The solid velocity at the interface can be split into its normal component and its tangential component. Finally, recalling that, at the micro time step at the interface, the continuity of fluid and solid normal velocities is enforced, equation (3.3) is rewritten as:

$$\mathbf{d}_{\Gamma_{FS}}^j = \mathbf{d}_{\Gamma_{FS}}^{j-1} + \Delta t_f \left((\mathbf{t}_s^{j-1} \cdot \mathbf{v}_{s_{\Gamma_{FS}}}^{j-1}) \mathbf{t}_s^{j-1} - (\mathbf{n}_f^{j-1} \cdot \mathbf{v}_{f_{\Gamma_{FS}}}^{j-1}) \mathbf{n}_f^{j-1} \right) \tag{3.4}$$

Hence, *free* conservative variables and *link* velocities are re-written such as:

$$\mathbf{U}^{j}_{f_{free_i}} = \frac{\Delta V_i^{j-1}}{\Delta V_i^{j}} \mathbf{U}^{j-1}_{f_i} - \frac{\Delta t_f}{\Delta V_i^{j}} \sum s_{ih}^{j-1} \tilde{\mathbf{F}}^{j-1}_{f_{ih}} \tag{3.5}$$

$$\mathbf{v}^{j}_{f_{link_i}} = \frac{\Delta t_f}{\rho^{j}_{f_i} \Delta V_i^{j}} \sum \mathbf{l}^{T}_{f_{ih}} \mathbf{\Lambda}^{j}_{ih} \tag{3.6}$$

To sum up, the algorithm of the monolithic co-simulation method Alg.2 is modified such as given in Alg.5.

Algorithm 5 MCS modified to backward Euler scheme

Initial fluid Ω_f and solid Ω_s states
Compute solid invariant : $\tilde{\mathbf{M}}_s, \mathbf{H}_s \leftarrow$ (2.95a)
while $t^m \leq T$ **do** ▷ Macro time loop
    Compute $\mathbf{U}^m_{s_{free}} \leftarrow$ (2.88)
    for $j \in [1, m]$ **do** ▷ Micro time loop
        Interpolate $\mathbf{v}^j_{s_{free}} \leftarrow$ (2.85b)
        ALE grid update $\mathbf{d}^j_{\Gamma_{FS}} \leftarrow$ (3.4)
        Compute $\mathbf{U}^j_{f_{free}} \leftarrow$ (3.5)
        Extrapolate $\mathbf{l}_f \big(\Delta V \rho\big)^j_{f_{free}}$ and $\mathbf{l}_f \mathbf{v}^j_{f_{free}}$
        Compute $\mathbf{H}^j_f \leftarrow$ (2.95c)
        Compute $\mathbf{\Lambda}^j \leftarrow$ (2.94b)
        Compute $\mathbf{v}^j_{f_{link}} \leftarrow$ (3.6)
        Compute $\mathbf{U}^j_f \leftarrow$ (2.87)
    end for
    compute $\mathbf{U}^m_{s_{link}} \leftarrow$ (2.92)
    compute $\mathbf{U}^m_s \leftarrow$ (2.87)
end while

The adapter developed to use OpenFOAM with the library is based on a callback function approach [24]. The adapter is an OpenFOAM function object (equivalent to UDF in Fluent): a shared library whose methods are called from predefined points in a solver's code. Function objects can be called, in the beginning of the first iteration of the time loop, during every iteration of the time loop and at the end of the last iteration of the time loop as example. Then, the adapter is loaded at runtime, using the existing controlDict configuration file, and the needed preCICE functions are executed according to the temporality of their function object.
The function object `read`, is called once at the beginning of the configuration. It reads the configuration and calls the preCICE function `initialize`.
The function object `execute` is executed at the end of every time or convergence iteration, after the fluid computation. It writes the coupling data, then calls the function `advance` and finally reads the coupling data.
The function object from the adapter `end`, calls the function `finalize`, closes the communication channels and frees the variables.

In this section, one CFD software, one dynamic solid software and a coupling library, all of them open-source, were presented. The chosen solvers from CalculiX and OpenFOAM use the same discretization methods as the methods presented in Chapter 2,

except for the fluid time integration scheme, where the explicit Runge-Kutta scheme is replaced by the explicit backward Euler scheme. Moreover, the coupling library preCICE can be used with these solvers to run FSI simulations, in a non-intrusive way. This library orchestrates the coupling simulation, manages the communication, the data transfer and mapping at the interface. The library also runs problems with different time scales even if the coupling data are not yet interpolated.
Thus, the idea is to integrate the proposed coupling method into preCICE in order to run FSI simulations with dedicated solvers, multi-time and space scales based on monolithic formulation and solved by a co-simulation algorithm. Fig.3.8 shows a representation of the coupling workflow that has to be implemented, where the ratio between the two time scales is $m = 2$.

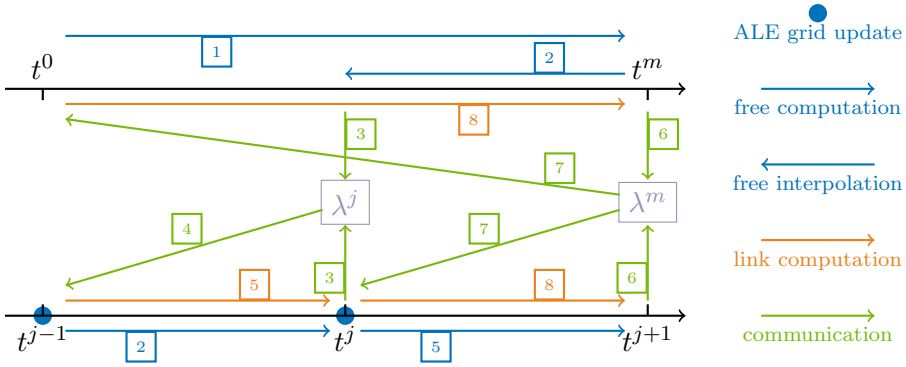


FIGURE 3.8: Preview of the MCS implementation workflow in preCICE with $m = 2$

The green arrows represent the exchanged data where the communication is handled by preCICE. Compared to a preCICE explicit coupling simulation, presented in section 3.1.1, the only feature which has to be modified from the library, is the coupling scheme. Fig.3.8 shows an optimized implementation of the MCS algorithm where several actions can be done in parallel, the boxed number being the order of execution. The implementation proposed below is fully sequential, but the prioritization order remains the same. The purple box represents the interface computation. This task can be executed by Python Action, a feature of the preCICE library which allows to execute python script at run time. Finally, the *free* states, blue arrows, and the *link* states, orange arrows, can be computed by the solvers into the adapters.
After this feasibility study, the implementation of the proposed method in this environment is presented in details into the next section.

3.2 Integration of the MCS algorithm into the library preCICE

In this section, the implementation of the proposed coupling method, called MCS for Monolithic Co-simulation, into the preCICE library and the CalculiX and openFOAM adapters, is detailed. The objective of the section is to describe in practice the implementation, but to remain at an architectural or algorithmic level to be as clear as possible. More details about the code implementation can be found in Appendix C.

3.2.1 Data exchanged

First, the data exchanges during the coupling computation are studied. The communication is led by TCP/IP sockets, data are exchanged from one adapter to the other by communication channels opened and closed by preCICE. Data fields are written or read on meshes from their respective solvers. Moreover, each mesh can hold only one field in writing and one field in reading. The exchanged fields are only variables projected on the fluid-structure interface.

Data written and read by the fluid adapter are now presented. First of all, fluid *free* normal velocities are required for the computation of interface operators, equation (2.94b). The FSI module of the openFOAM adapter does not allow the writing of velocities. Then, following the model of the script `Displacement.C`, the script `Velocity.C` is created, with a function that writes the normal velocity at the interface nodes into the buffer, see C.2. This variable is defined in preCICE by the mean of the `XML` configuration file of the simulation, as a vector called `nVelF` defined on `Fluid-Mesh-Nodes`, the nodal fluid mesh of the interface.
A second field required in writing for the MCS coupling is the values of the fluid mass at the interface, $\left(\Delta V \rho\right)^j_{f_{\Gamma_{FS}}}$, in order to compute the fluid operator $\mathbf{H}^j_f$, equation (2.95c). Then, the script `Mass.C`, see C.3, is implemented to create a field, located at each cell's center, which is the mass of the adjacent cells of the interface, using their volume and density. Then they are projected to the cells' face included in the interface. This field is defined for preCICE as the scalar `massF` written on the mesh `Fluid-Mesh-Faces`.
In reading, the first field of interest is the Lagrange multipliers which are used to compute the *link* velocities of the fluid equation (3.6), and then, the total fluid state (2.87). In this way, the function `read` from `VelocityLink.C` script is called in the adapter by `ReadCouplingData`, see C.4. Hence, the forces of interaction are read at each cell's faces, from the buffer. Then the velocity at the center of the cells, whose at least one face is included into the interface, is modified, adding the *link* velocity term. At the preCICE configuration, Lagrange multipliers are vectors named `lambda` read at the meshes `Fluid-Mesh-Faces`, as for imposed forces or pressure in fluid computation based on FVM.
Finally, the last required field for the openFOAM adapter, is the tangential solid velocity at the interface, used for the computation of the ALE fluid grid, equation (3.4). The vector `tVelS`, read from the mesh `Fluid-Mesh-Nodes`, is used into `GridDisp.C` to compute, with the normal fluid velocities, the nodal displacements of the interface, see Appendix C.5.

Concerning the solid sub-domain, the CalculiX adapter writes, as for the fluid, the normal velocities vectors, called `nVelS`. They are defined at the node of the solid mesh, `Solid-Nodes1`. The writing of the tangential velocities vectors, `tVelS`, is also required. These values are also computed at the node of the solid mesh. Nevertheless, only one write per defined coupled mesh is allowed. Thus, a second instance of the solid mesh, `Solid-Nodes2` is created, where the tangential solid velocities are written. Finally, a third exchange field and a third instance of the solid mesh have to be created; they are not directly used for the solid sub-domain computation but mandatory for the implementation of the exchanged fields of the MCS scheme. These void fields have to be three dimensional vectors. Then `zero` are written at `Solid-Mesh3`. The solid solver CalculiX uses only one nodal mesh, but three instances of this one are defined in the coupling configuration due to the need of writing three different fields

for the coupling. This written fields were not available into CalculiX adapter. For this purpose, they have been added, see Appendix C.6.
In reading, the CalculiX adapter requires the interface forces, in order to compute the solid *link* vector state. Thus the preCICE configuration defines the vectors `Lambda` at the mesh `Solid-Nodes1`. This field is read by the adapter using the existing function `getNodesForces`.

Henceforth, variables that have been exchanged are defined from the adapter to their meshes, and the communication procedure is presented. First, as mentioned in section 3.1.1, the communication between the two solvers is run by the transfer of data from the same mesh, available into the fluid participant or into the solid participant. Here we choose to define solid meshes, into the fluid participant, due to the fact that the fluid sub-domain is run by the finer time scale. Then, six exchanged fields have been introduced but only five communication channels are created. Indeed, thanks to pythonAction, which can modify the value of exchanged data at run time, two channels use two different fields during the communication. Fig.3.9 shows a representation of the data exchange used for the proposed coupling. The blue and orange arrows represent the variables written to and read from the adapters fluid and solid respectively. Mappings are illustrated with black arrows while data transfers are shown by green arrows. When a variable is modified during its exchange, the variable's name is made of the two fields names separated with an underscore. The name of the current field, contained in the variable is highlighted in color. Recalling that communication is lead from one adapter to the other, the five data exchanges are described: the fields they contain and where they are written, with no temporality consideration.

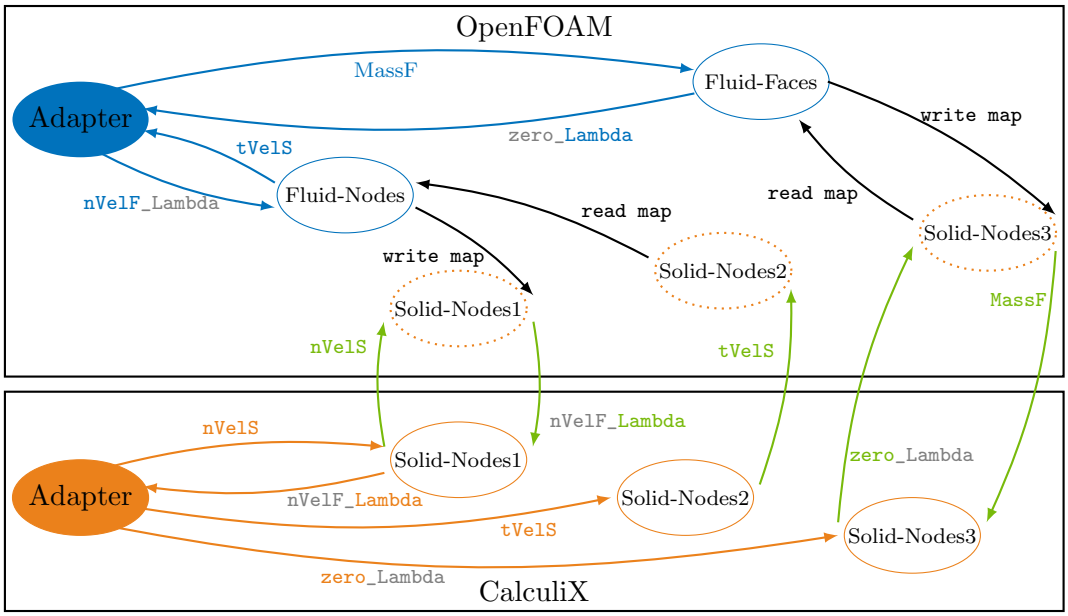


FIGURE 3.9: Exchanged data diagram for MCS coupling

First, the solid normal free velocities `nVelS` are written from the CalcluliX adapter to the mesh `Solid-Nodes1`. Then, they are transferred to the mesh `Solid-Nodes1` of CalculiX, where they are used for the interface computation but the exchange is stopped here.
Also, the OpenFOAM adapter writes the normal free velocities into `nVelF_Lambda`, at the mesh `Fluid-Nodes`. Then they are mapped to the meshes `Solid-Nodes1` on

the fluid side. At this point, the Lagrange multipliers are computed, and their values replace the fluid normal velocities into `nVelF_Lambda`. Finally, they are transferred to the mesh `Solid-Nodes1` of the solid participant. And the Lagrange multipliers are read by CalculiX adapter.
The OpenFOAM adapter also writes `MassF` at `Fluid-Faces`, which is mapped on `Solid-Nodes3` from the fluid participant. Data are finally transferred to the Solid participant even if they are not used into the CalculiX adapter, the `write_coupling_data` function of the adapters being called by the preCICE data transfer.
The CalcuiX adapter fills `Zero_Lambda` with zero vectors. Then this data is transferred from the solid participant to the fluid participant through the mesh `Solid-Nodes3`. Then the exchange variables are replaced by the fields of the interface interaction forces, and mapped to the mesh `Fluid-Faces` where they are read by the openFOAM adapter. Finally, the last communication channel is for `tVelS`, written by CalculiX adapter to `Solid-Nodes2` and transferred to the fluid participant before they are mapped to `Fluid-Nodes` and read by the fluid adapter to update the interface displacements.

3.2.2 Python action

As said previously, several operations occur during the data exchange to modify variables at runtime. These modifications occur by the mean of Actions, Python scripts written by the user and executed at runtime. They are defined at a mesh and can access the available data on this mesh, the time step and time. They can be executed at different moments of the coupling simulation, before or after mapping is read or written, or at the end of the coupling step. The Action is configured into the `XML` file of the coupling simulation. Four Actions are used during the implementation of the proposed coupling method, which are available in Appendix C and their functioning are described below.

The first Action is called `computeHf.py`. It is defined on mesh `Solid-Nodes3` from the fluid participant and is run at each micro time step, after the mapping of `massF` between `Fluid-Faces` and `Solid-Nodes3` has been performed. The function `performAction` of this script, uses the time step and the `massF` variable to compute the fluid operator $\mathbf{H}_f^{j+1}$, equation (2.95c), at the solid nodes. Then this two dimensional numpy arrays, are written into a numpy binary file `Hf.npy`.

The most complex and important Action is `computeLambda`, which solves the interface problem. It is defined in the fluid participant, on the mesh `Solid-Nodes1` and its `performAction` function is called after the mapping of `nVelF_Lambda` from `Fluid-Nodes` to `Solid-Nodes1`.
If it is initialization, the operator $\tilde{\mathbf{M}}_s$, written beforehand into a `.txt` file by the CalculiX adapter, is loaded and used to compute the solid interface $\mathbf{H}_s$ following (2.95a). Then, it is saved into `Hs.npy`.
After that, at each time step, interface operators are loaded from `Hf.npy` and `Hs.npy` and used with the normal *free* velocities, solid and fluid both available on this mesh, to compute the interaction forces, equation (2.94a). If needed, the solid *free* normal velocity is first interpolated at the micro time step. Finally, Lagrange multipliers are saved to `Lambda.npy`, but also written into `nVelF_Lambda` instead of the *free* fluid velocities.

The Action `TransferData.py` is used to allow the openFOAM adapter to also read the interface interaction forces. Thus, this action is defined on the mesh `Solid-Nodes3`, before the mapping of the data `Zero_Lambda` to the *Fluid-Faces* mesh. In this way, `Zero_Lambda` can get the values of the Lagrange multipliers loaded from `Lambda.npy`, which had been previously written at the same mesh.

The last Action called `computeMicro`, is defined before the reading of the mapping from the mesh `Solid-Nodes2`, in order to interpolate the tangential solid velocities at the previous fluid time step t^{j-1}, if two different time scales are used. The variable `tvelS` then receives the value of the interpolation. This last one is used for the ALE grid interface correction, see equation (3.4).

3.2.3 Temporality of the coupling scheme

The implementation of the proposed coupling scheme algorithm is based on the serial explicit scheme from preCICE. A new coupling scheme tag is created into the coupling library `MCS`. If this one is used in the coupling configuration file, the coupling is initialized as an explicit coupling scheme. Then, thanks to the defined exchanged fields, the Actions and the modification of the adapters, the MCS coupling algorithm is run instead of the serial-explicit coupling algorithm. Also, the first participant has to always be the CalculiX participant and the coupling time step has to be defined by the method `first-participant`. In case of two different time scales, with macro scale for solid integration and micro scale for fluid integration, the coupling data are transferred only at the macro time step.

Some modifications have been made for the CalculiX adapter comparing to its algorithm 4. First, at the initialization of the simulation, the effective mass matrix $\tilde{\mathbf{M}}_s$ is written into a `.txt` file, in order to be used in the action `computeLambda.py` to construct the solid operator $\mathbf{H}_s$. Then, the logical variable `Precice_isMcsCouplingScheme` is created at the coupling set up. It is used to modify the adapter coupling scheme, in particular when reading and writing of the coupling data occur. Recalling that for serial explicit scheme, at each time step, coupling data are read. Then the solid is solved using coupling data as Neumann boundary condition. Finally, the coupling data from this computation are written into the buffer to be sent to the other participant, while the CalculiX adapter moves to the next time step. During the MCS coupling scheme, coupling data are read after the preCICE `advance` function. Also, the *link* computation is added, see details in C.7. Algorithm 6 gives the algorithm of the modified CalculiX adapter.

Concerning the OpenFOAM adapter, such modifications do not have to be implemented due to the fact that the function object `execute`, which is called after fluid computation, calls the writing of the coupling data, then `advance` preCICE function and finally the reading of the coupling data. The only missing instruction to complete the MCS procedure is the `link` computation and the grid correction which are directly implemented into the coupling data reading functions, see respectively C.4 and C.5.

Algorithm 6 CalculiX adapter with MCS implementation

```
1: ccx initialization
2: write M̃_s
3: Precice_Setup                                   ▷ initialize is called
4: while Precice_IsCouplingOngoing do              ▷ Temporal loop
5:     Precice_AdjustSolverTimestep
6:     if Precice_isMcsCouplingScheme == False then
7:         Precice_ReadCouplingData
8:     end if
9:     if Precice_IsWriteCheckpointRequired then   ▷ for implicit coupling
10:        Save checkpoint
11:    end if
12:    ccx solid computation
13:    Precice_WriteCouplingData
14:    Precice_Advance
15:    if Precice_isMcsCouplingScheme == True then
16:        Precice_ReadCouplingData
17:        ccx Link computation
18:    end if
19:    if Precice_IsReadCheckpointRequired then    ▷ for implicit coupling
20:        Load variables from checkpoint
21:    else
22:        end time step
23:    end if
24: end while
25: PreCICE_FreeData                               ▷ finalize is called
```

Finally, Fig.3.10 shows the global time loop of the fluid-structure MCS coupling orchestrated by preCICE. The grey instructions are executed by the solvers. Orange and blue instructions are respectively executed by CalculiX and OpenFOAM adapters. Data transfers are shown in green, mapping in black and Action in purple; all of them being performed by preCICE.

This section presented the implementation of the coupling method proposed in Chapter 2, to couple CalculiX and OpenFOAM to run FSI simulations using the preCICE library. Some functionalities have been added regarding exchanged fields and the coupling algorithm, only the coupling library and the adapters have been modified, solvers were not. The configuration required to run such a coupling problem, see AppendixC.1, shows a good overview of the implementation operating.

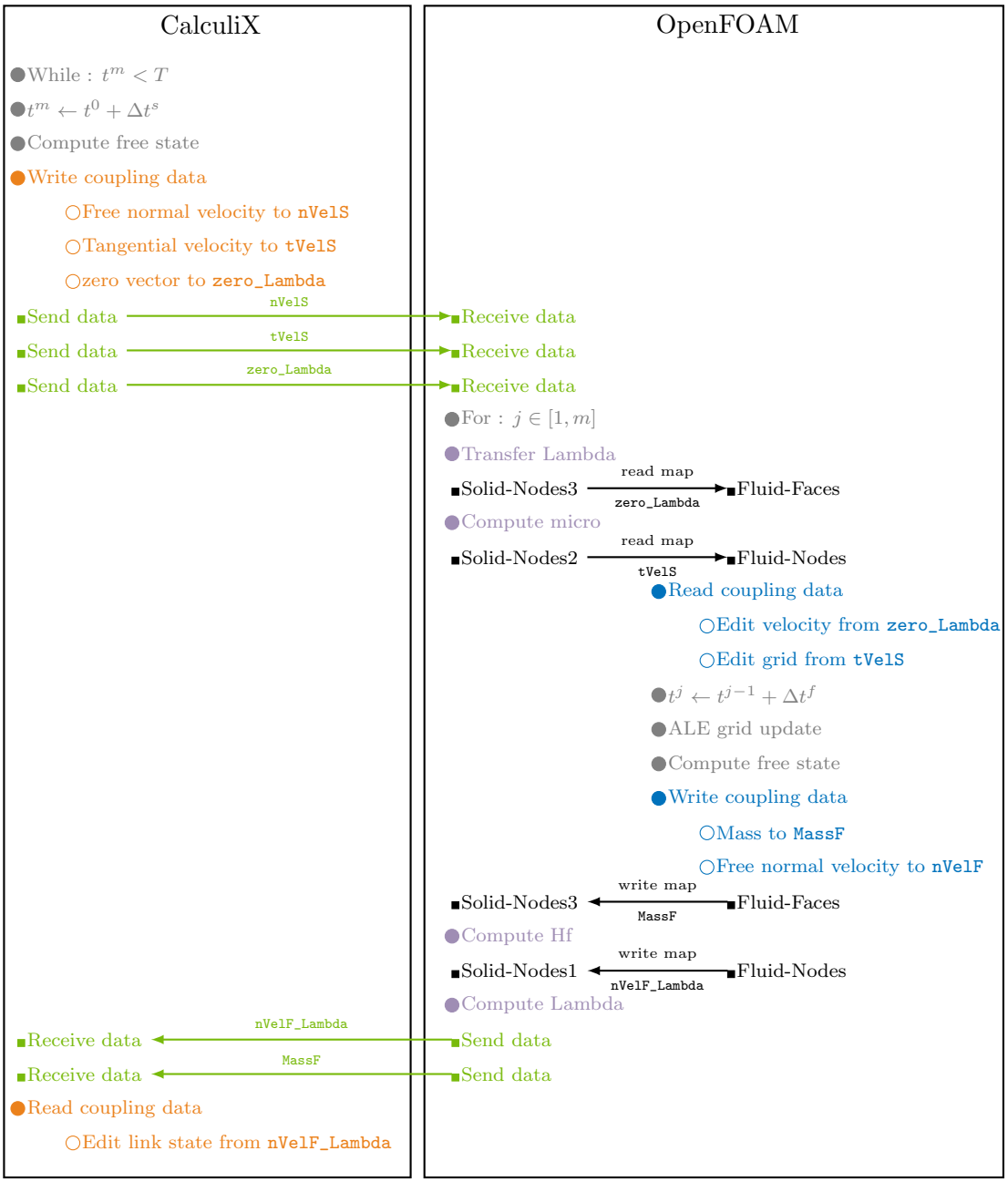


FIGURE 3.10: Temporal resolution of MCS coupling in preCICE environment

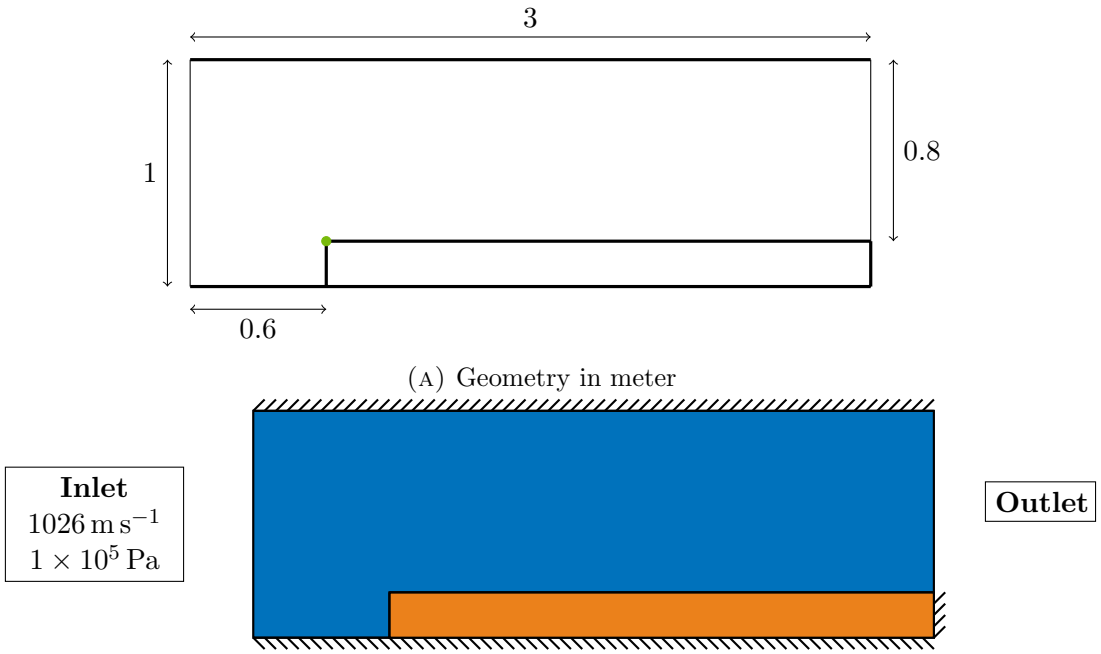
3.3 Numerical test cases

Now the MCS method is available in the preCICE environment, this one can be used to solve FSI simulation using CalculiX and OpenFOAM for sub-domains computation. Both solvers are three dimensional solvers. In this way, the presented problems are solved in three dimensions even if they come from two dimensional benchmark. They are mapped in three dimensions, but in the z direction, there is only one layer of finite elements or finite volumes to discretize the solid and the fluid sub-domains, respectively. Then, symmetry boundary conditions are enforced on the front faces and the back faces. The two FSI test cases chosen to test the MCS method are the *forward step* problem and the *perpendicular flap* problem, where different physics of

fluid-solid interaction are involved. The following section presents the test cases and the numerical results issued from MCS co-simulation.

3.3.1 Forward step

The first test case studied, is the *forward step* problem. This test case has been introduced in [36] as a fluid supersonic test case, which is extended here to fluid-structure interaction simulation. We consider a wind tunnel with a step which is the solid sub-domain. The wind tunnel is 1 meter wide and 3 meters long, full of air, at atmospheric pressure. The inlet is a uniform flow at Mach 3 velocity, $1023\,\mathrm{m\,s^{-1}}$. The step is located at 0.6 meter from the inlet and is 0.2 meter high with a density of $7800\,\mathrm{kg\,m^{-3}}$ and Young's modulus of $2 \times 10^{11}\,\mathrm{N\,m^{-2}}$. Fig.3.19 presents the geometry and boundary conditions of the considered problem. The green dot in Fig.3.19a shows the localisation of the interest point where the displacements of the interface are studied. Thus, this test case treats of a stiff but deformable solid sub-domain, and an high speed velocity flow. These hypotheses are often used for aerospace FSI simulation for example.

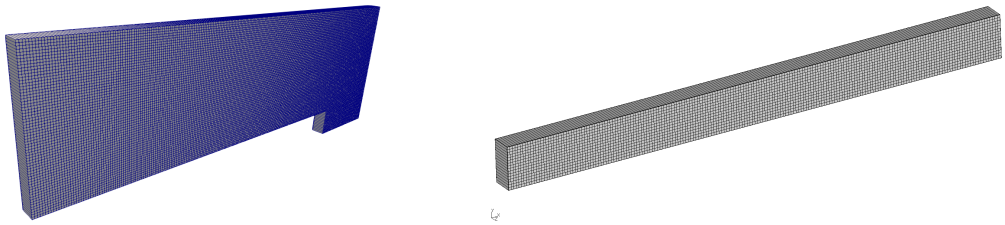


(A) Geometry in meter

(B) Boundary conditions

FIGURE 3.11: Definition of the forward step FSI problem

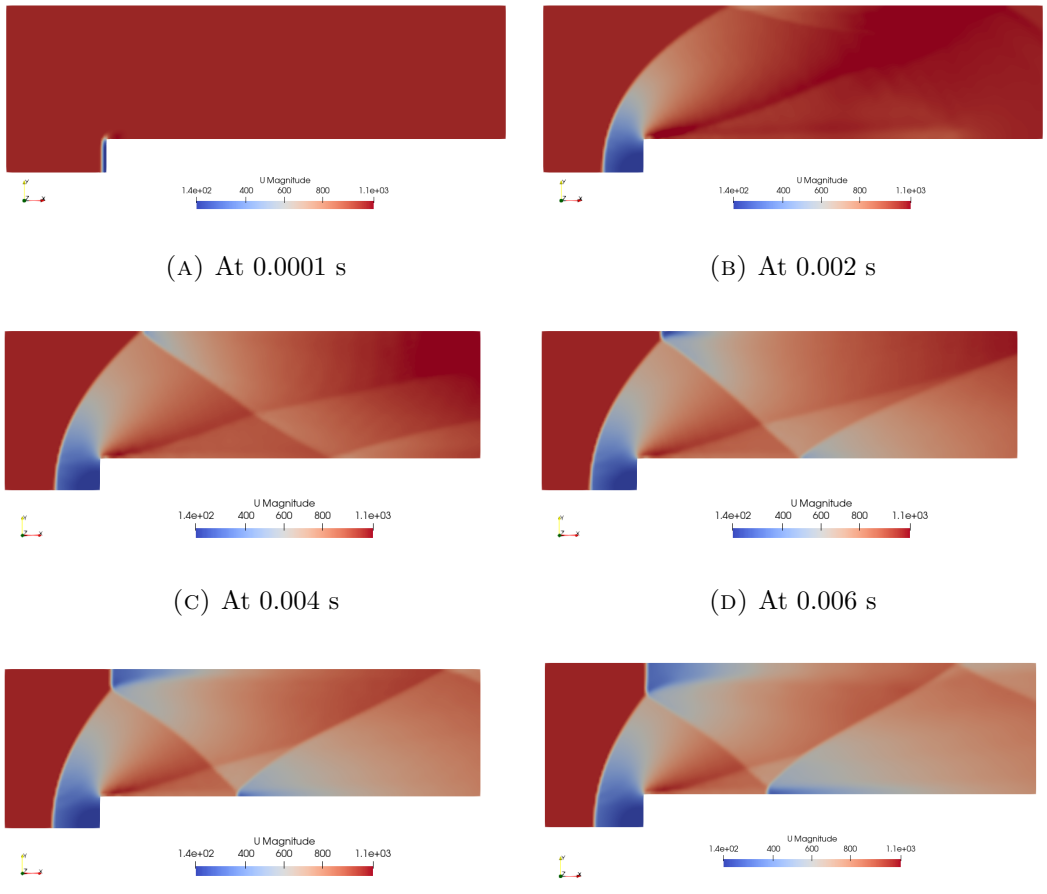
Concerning the spatial discretization, the nodes of both meshes are consistent at the fluid-structure interface. The Fig.3.12 presents the discretized sub-domains which are made of 16000 finite volumes for the fluid sub-domain and 3072 finite elements for the solid sub-domain.
The first simulation run is mono scale in space and time. The time step used over both subdomains is $\Delta t_s = \Delta t_f = 1 \times 10^{-6}\,\mathrm{s}$, which is inferior to $3 \times 10^{-6}\,\mathrm{s}$, the critical time step, induced by the explicit time integration scheme over the fluid sub-domain. The problem is computed from $0\,\mathrm{s}$ to $T = 0.01\,\mathrm{s}$.



(A) Fluid sub-domain mesh (B) solid sub-domain mesh

FIGURE 3.12: Three dimensional forward step case conforming meshes

The first results presented are the velocities into the fluid sub-domain. Fig.3.13, shows the magnitude of the flow velocities at six successive times. The results are consistent and really close to the one way simulation of the problem, where the solid sub-domain is considered as a rigid wall. The main difference is that for the mono scale MCS coupling, the computed velocities at the Fluid-Solid interface are slightly higher.



(A) At 0.0001 s (B) At 0.002 s

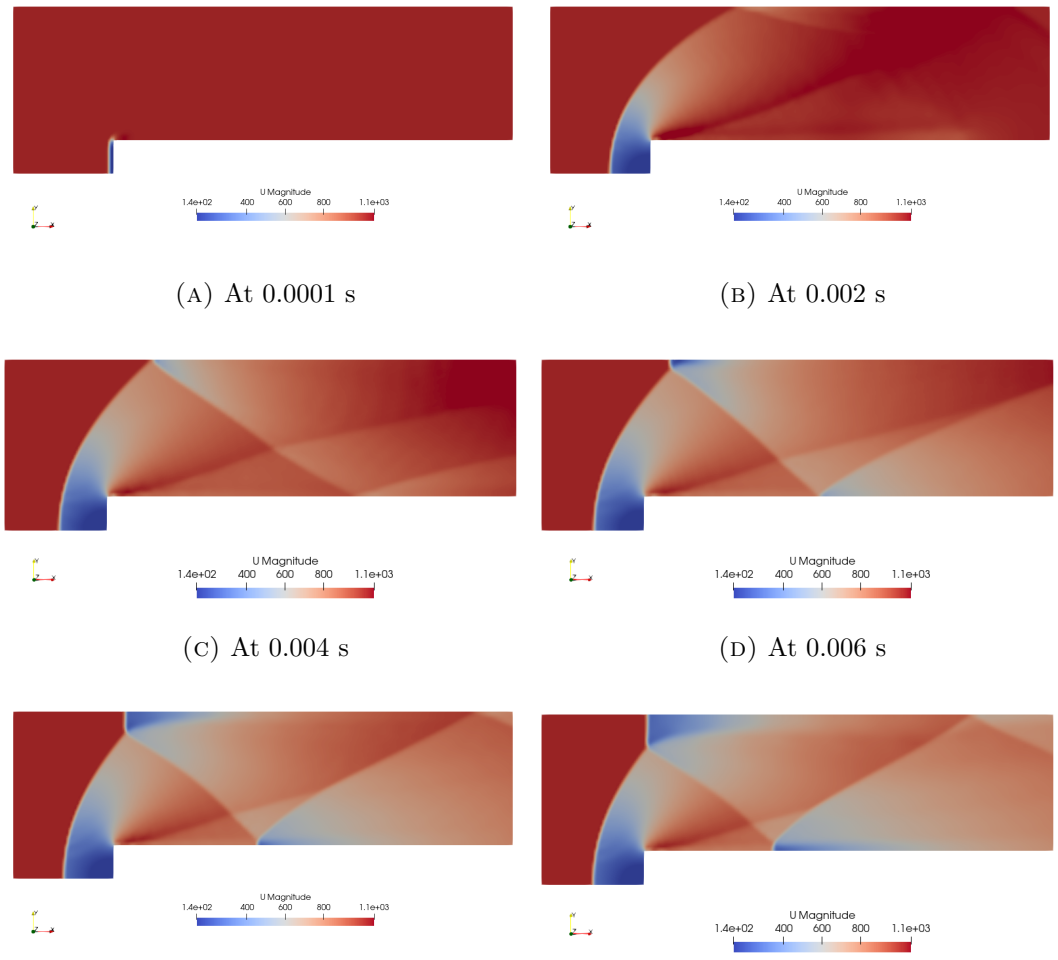
(C) At 0.004 s (D) At 0.006 s

(E) At 0.008 s (F) At 0.01 s

FIGURE 3.13: Velocity magnitude of the fluid sub-domain at different times, with mono time-space scale and conforming meshes

During, a second phase, the simulation is recomputed with the same meshes but using different time steps for each solver. The fluid sub-domain uses the thinner time scale,

$\Delta t_f = 1 \times 10^{-6}$ s as previously, whereas the solid sub-domain is computed less often, using a larger time step such as $\Delta t_s = m\Delta t_f$. Fig.3.14, shows the same results as Fig.3.13 in terms of fluid velocities, but these results have been obtained during a simulation with a time step ratio of 10 between the two sub-domains. Despite this main difference between the two co-simulations, the results in velocity are very consistent between each other.



(A) At 0.0001 s (B) At 0.002 s

(C) At 0.004 s (D) At 0.006 s

(E) At 0.008 s (F) At 0.01 s

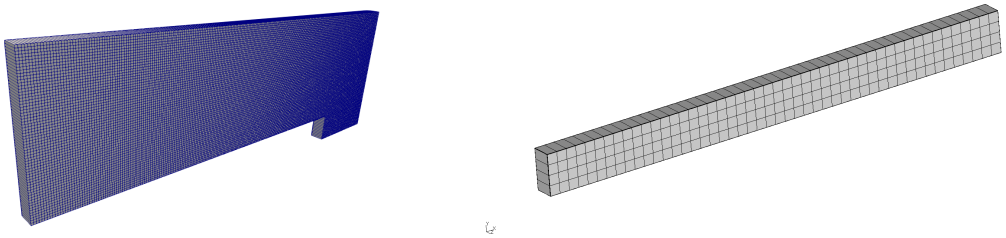
FIGURE 3.14: Velocity magnitude of the fluid sub-domain at different times, with time step ratio $m = 10$ and conforming meshes

To investigate the performance of the multi-time scale resolution, which exhibited accuracy degradation with the increase of the ratio m in the previous chapter, the displacements of a point of interest are studied. The chosen point is the corner of the solid domain, shown by the green dot in Fig.3.19a. Thereby, Fig.3.15, shows the displacements of this point for a mono-time scale simulation, in blue, and for multi-time scale simulations with $m = 5$, in orange, and $m = 10$ in green.
The displacements for each simulation are in the same range, but the differences of the results are more obvious than for the velocities results in the fluid sub-domain. For the computation with $m = 5$, the displacements are close to the reference values, the mono time scale simulation, but are a bit under estimated. For the results with a solid time step ten time larger than the fluid time step, the displacements of the

considered point of the interface seem to be a phase shifted compared to the reference, probably induced by a weak loss of energy. Nevertheless, the results seem to be slightly impacted by the increase of the time step ratio, in the frame of conforming meshes at the interface.

FIGURE 3.15: Displacements of the point of interest according to the time, with different time scale ratio and conforming meshes

The physics of the test case and the integration scheme imposed a mesh with small elements for the fluid-domain which is not necessary for the linear elastic solid sub-domain. In order to run a coupling simulation with appropriate time and spatial scales, used for each sub-domain, a second mesh is proposed, see Fig.3.16. For this simulation the solid mesh is four times larger than the fluid mesh. In this way, meshes at the interface are not conform.

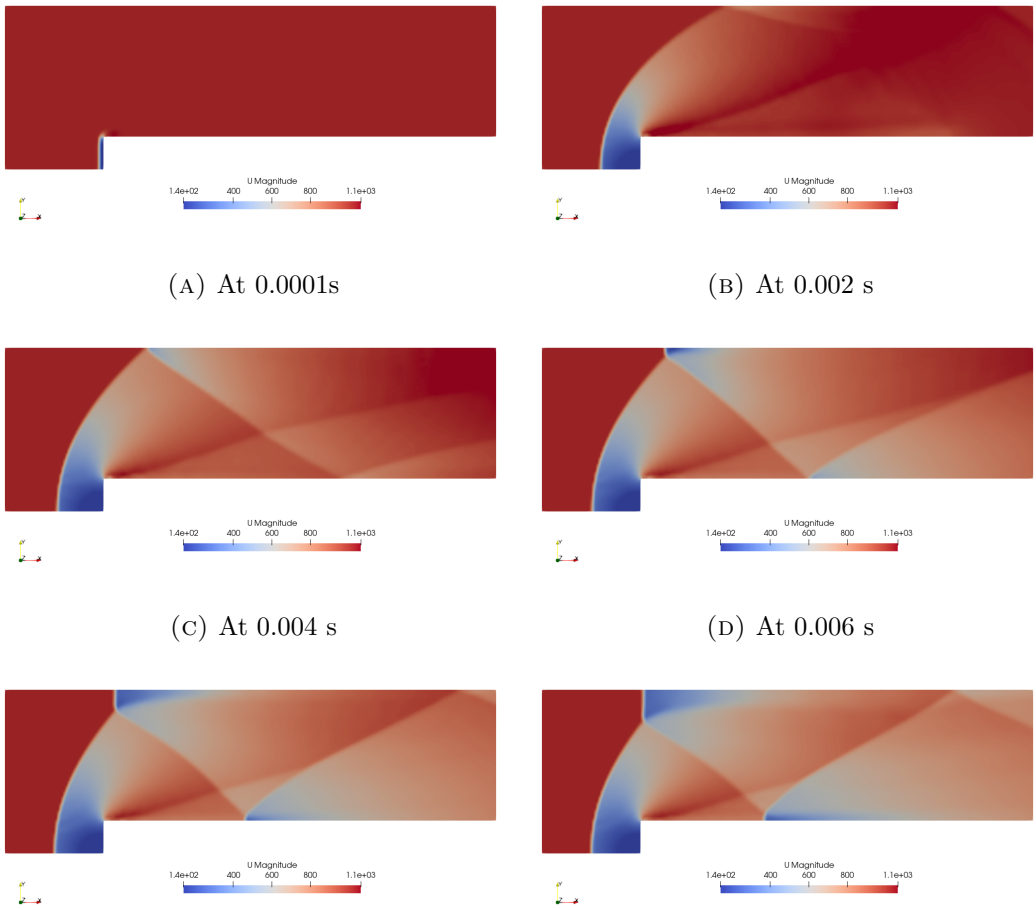


(A) Fluid sub-domain mesh

(B) Solid sub-domain mesh

FIGURE 3.16: Three dimensional forward step case non-matching meshes

Then, the simulation is first run with the same time step used by the fluid and the solid solvers. The results in terms of velocity inside the fluid sub-domain are visualized in Fig.3.17 and show good similarities in comparison to the previous results.



(A) At 0.0001s (B) At 0.002 s

(C) At 0.004 s (D) At 0.006 s

(E) At 0.008 s (F) At 0.01 s

FIGURE 3.17: Velocity magnitude of the fluid sub-domain at different times, with mono time-space scale and non-conforming meshes

Finally, a multi-time and space scale simulation is run, with $m = 10$. Fig.3.18 presents the displacements of the point of interest. The dotted lined represents the results discussed previously for a simulation with conforming meshes while the solid lines are the results obtained by the computation of the problem with the larger mesh for the solid. The blue lines concern mono time scales simulation, while the orange lines are computed with $m = 10$. As for the increase of the time step ratio, the non-conforming meshes simulation shows results in displacement a bit different from the referential solution. But the differences seem to be reasonable regarding the performance increase with the use of appropriate time and space scales.

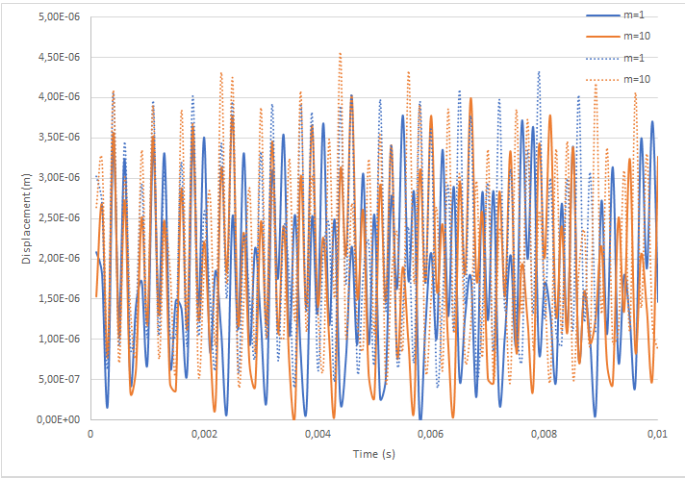
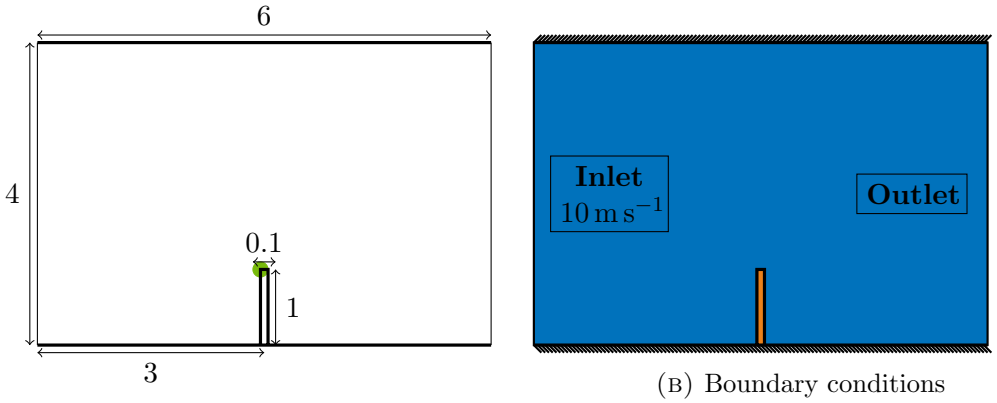


FIGURE 3.18: Comparison of the displacements of the point of interest according to the time, for conforming (dotted lines) and non conforming (solid lines) meshes, with $m = 1$ and $m = 10$

3.3.2 Perpendicular Flap

The second test case used to validate the implementation of the proposed method, is the well-known FSI problem of the *perpendicular flap*. The fluid domain is constituted of a channel of air, with a velocity of $10\,\mathrm{m\,s^{-1}}$ at the inlet. In the center of the bottom of the fluid sub-domain, a flexible flap, perpendicular to the flow is embedded. This flap, is the solid sub-domain for FSI simulation, with Young's modulus of $4 \times 10^6\,\mathrm{N\,m^{-2}}$ and a density of $3000\,\mathrm{kg\,m^{-3}}$. The geometry and dimension parameters of the problem are shown in Fig.3.19a. Once again, a point of interest is chosen and indicated by a green dot, at the top left corner of the flap. This test case can be assimilated to a biomedical simulation, where low velocity flows and important deformations of the solid can occur.



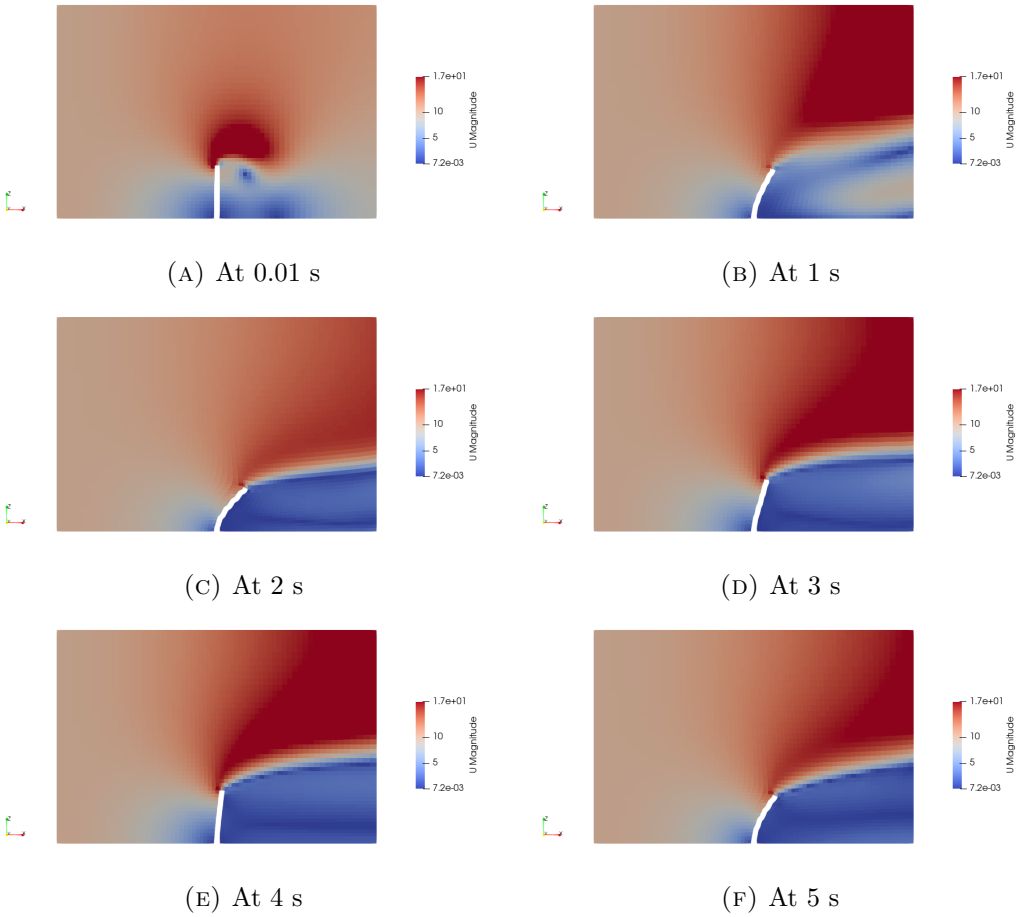
(A) Geometry in meter

(B) Boundary conditions

FIGURE 3.19: Definition of the forward step FSI problem

As in the previous test case, first a mono-time scale co-simulation is considered, with $\Delta t_s = \Delta t_f = 1 \times 10^{-5}\,\mathrm{s}$, which is inferior to the critical time scale involved for fluid computation of $2 \times 10^{-5}\,\mathrm{s}$. The simulation is run from $t^0 = 0\mathrm{s}$ to $T = 5\mathrm{s}$. Fig.3.20, gives the magnitude of the fluid velocity at different successive times. In the following,

these results will be considered as reference results for comparison purpose.



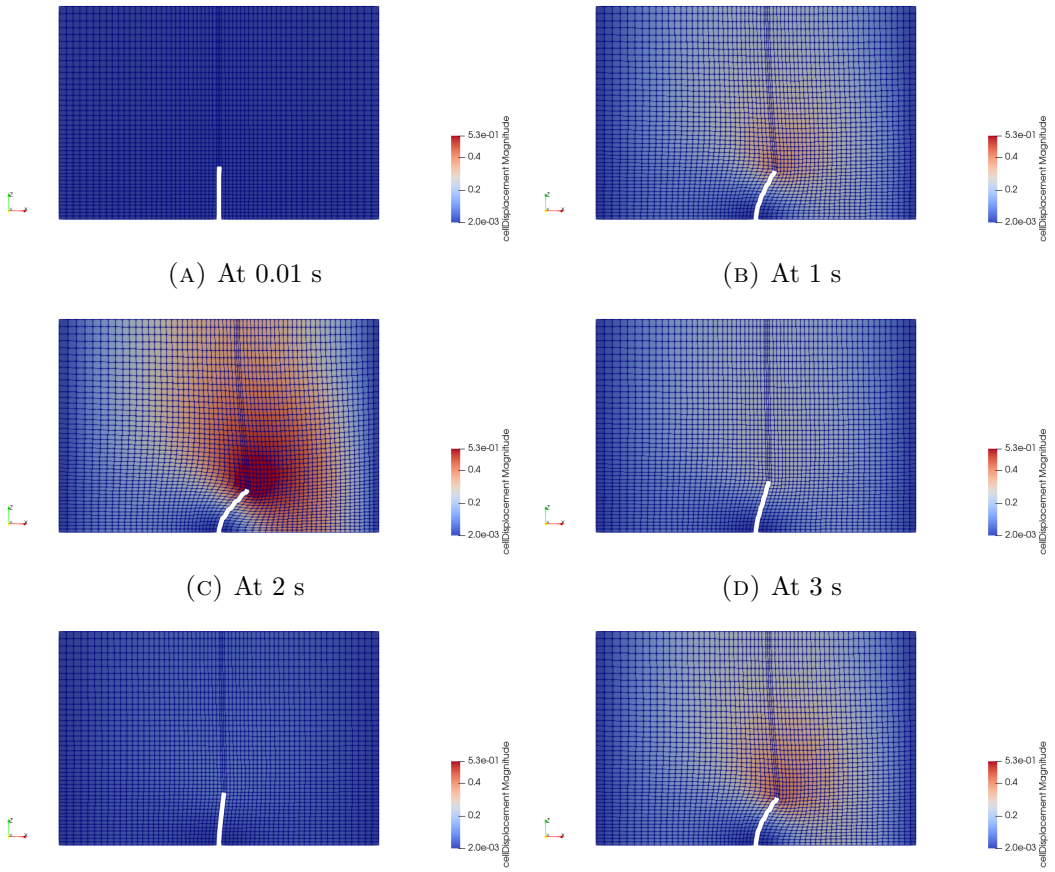
(A) At 0.01 s (B) At 1 s

(C) At 2 s (D) At 3 s

(E) At 4 s (F) At 5 s

FIGURE 3.20: Velocity magnitude in the fluid sub-domain at different times, obtained with mono time-scale co-simulation

The Fig.3.21 allows to visualize the ALE grid deformation at different times for the mono-time scale co-simulation. We can see, comparing to the velocity, that the ALE grid velocities is well updated according to time and does not follow the fluid velocities, enabling us to prevent severe distortion in the fluid mesh. Moreover, at the fluid-solid interface, the fluid mesh is deformed to fit the solid domain motion, the fluid mesh is being considered as Lagrangian at the boundaries, which simplifies the data transfer orchestrated by the coupling library preCICE.



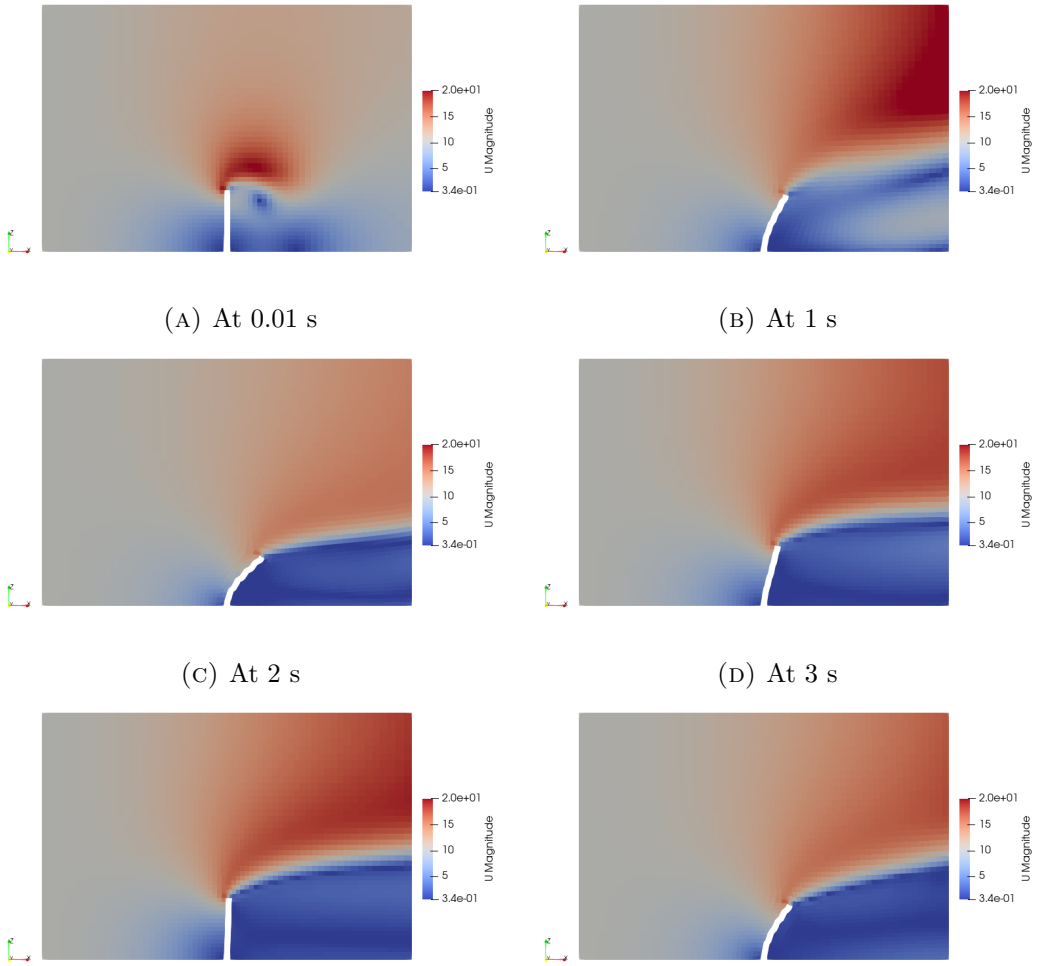
(A) At 0.01 s (B) At 1 s

(C) At 2 s (D) At 3 s

(E) At 4 s (F) At 5 s

FIGURE 3.21: Displacement magnitude of ALE grid at different times, obtained with mono time-scale co-simulation

Finally, the problem is computed using dedicated time scales depending on sub-domains. Then, the co-simulation is performed, where the solid time scale is five times and ten times larger than the fluid time scale. Fig.3.22 presents the velocities of the fluid flows for $m = 10$. Again, the results inside the fluid sub-domain are close to the reference ones.



(A) At 0.01 s (B) At 1 s

(C) At 2 s (D) At 3 s

(E) At 4 s (F) At 5 s

FIGURE 3.22: Velocity magnitude of the fluid sub-domain at different times, obtained with multi time-scales, $m = 10$, co-simulation

In order to study the potential degradation of the results induced by the increase of the time step ratio, variables of interest are studied at the fluid-solid interface. In this way, the x component of the displacement at the point of interest is plotted in Fig.3.23, for several co-simulations.
First, the co-simulation has been computed with the explicit serial coupling scheme with mono time scale, based on partitioned formulation, available in the preCICE library. The results are shown by the purple line. The loss of amplitude is quite important.
Then this result is compared to mono-time scale co-simulation run by the MCS scheme implemented in the preCICE library. The displacement according to x are plotted with the blue line, its amplitude remains stable.
Finally, MCS multi-time scales cases are studied. The dashed orange and dotted green results are the displacements in the x direction of the tip of the flap for the computation with $m = 5$ and $m = 10$ respectively. A very weak loss of amplitude occurs with the multi-time scale simulation. This loss is larger for $m = 10$ than for $m = 5$, remaining very reasonable, compared to the total amplitude. Also, it is really smaller than the results from the serial explicit mono-time scale resolution even when $m = 10$.

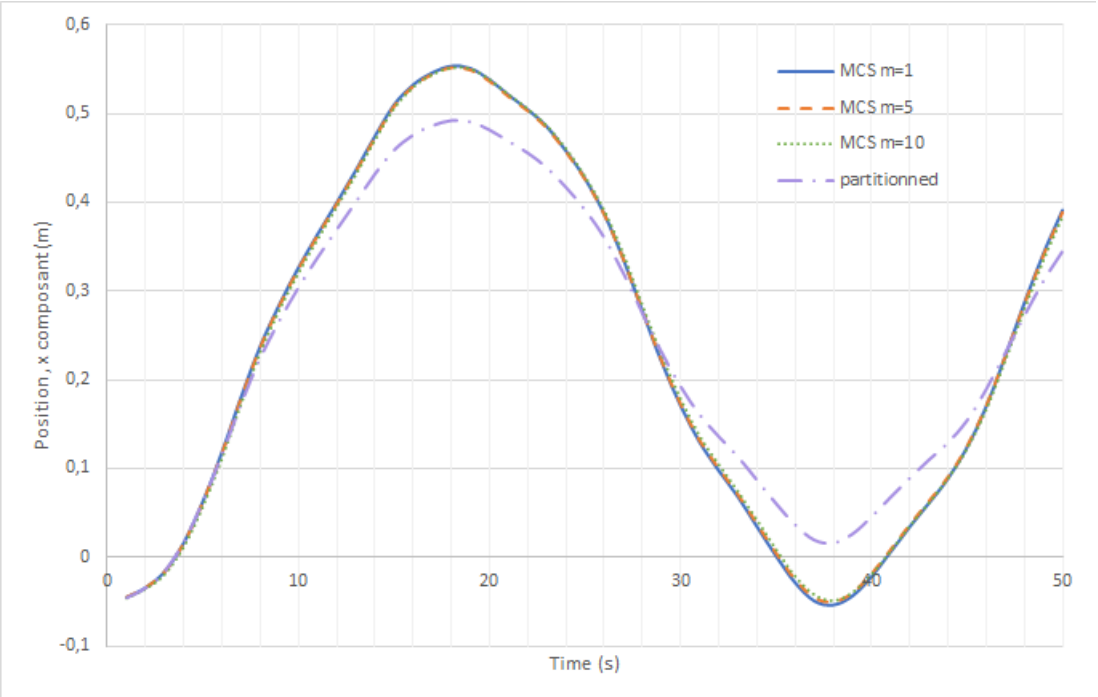


FIGURE 3.23: x component of the position of the point of interest, for MCS coupling with different time scales and mono time scale partitioned serial explicit coupling

The MCS results from this test case are convincing, and confirm the relevance of the proposed approach detailed previously in the piston problem, in Chapter2.

Conclusion

In this chapter, the problems of the implementation method have been discussed, and the chosen software have been presented. Then, the architecture and the orchestration developed for the integration of the method were detailed. Finally, the implementation was tested with two different FSI benchmarks: aerospace and biomedical. The implementation allows running simulations at multiple time and space scales. The results of the variables of interest at the fluid-structure interface can be slightly degraded. Nevertheless, the results inside the sub-domains remain close to the reference results. Therefore, computation time can be saved in return for a reasonable loss of accuracy.

General conclusion and perspectives

This thesis presents the work carried out to develop a new method of coupling for transient FSI problems, in order to couple dedicated solvers, using their own discretization methods, as less intrusively as possible. In space, fluid and solid sub-domains use different discretization methods, and the meshes at the interfaces can be incompatible. Regarding time, different integration schemes are employed and each solver is computed with a dedicated time scale. Generally speaking, the proposed method is flexible, stable and based on energy preservation.

The first chapter, proposed a review of the existing coupling approaches for FSI simulation. We proposed to divide FSI problems according to their formulations into partitioned or monolithic approaches. On the one hand, partitioned couplings propose to solve FSI problems, computing fluid and solid sub-domains separately. They are flexible but can lack stability and accuracy. On the other hand, monolithic approaches propose to solve each sub-domain synchronously, which is more stable and accurate but in return is more difficult to implement and rarely generalizable. Finally, this chapter presented domains decomposition methods for solid problems and then their extension to FSI simulation. These approaches are really attractive because they allow to combine the advantages of partitioned and monolithic couplings. For all of these reasons, the choice was made for the proposed coupling method, to be based on a Schur dual monolithic formulation, with an interface tracking approach, for stability and accuracy, solved by a co-simulation algorithm based on the GC method, for flexibility.

In the second chapter, the discretization methods used for each sub-domain and the proposed coupling method were presented. The solid sub-domain, considered continuous and linear elastic, is spatially discretized using FEM in Lagrangian formulation and is temporally integrated by an implicit Newmark scheme. Concerning the fluid sub-domain, it is considered inviscid and compressible, written with an ALE formulation. It is discretized using cell centered finite volume method and an explicit Runge-Kutta of order two scheme. For the FSI simulation, each sub-domain uses its own time step, and the fluid sub-domain uses the thinner time scale, due to the explicit time integration. Then, the coupled problem is solved using a *free*/*link* decomposition based on the GC method. Compared to a standard serial explicit coupling scheme, the proposed MCS scheme is a bit costlier due to the computation of the interface problem. Nevertheless, concerning the simulation of the piston test case presented at the end of the chapter, its results are really more accurate. First, the error of the normal velocities continuity at the interface is around the computer accuracy. Moreover, the coupling is energy conservative in case of a mono time scale simulation. A reasonable loss of energy is noticed for multi-time scale coupling. Nevertheless, concerning the variables of interest at the interface, as velocities or displacements, the

results are slightly degraded, by the increase of the ratio between the fluid and the solid time steps. There is no loss of amplitude at all for mono-time scale while this loss is important for the serial explicit coupling.

Finally, the last chapter was dedicated to the implementation of the method into a free environment, in order to be run with dedicated CFD and solid dynamic solvers, thus allowing the computation of various and more complex FSI problems. The choice was made to use the library preCICE; a coupling library which allows the coupling of existing solvers to run FSI simulation according to several partitioned coupling schemes, managing the communication and the data mapping. The solid sub-domain is computed by the free software CalculiX, while the fluid sub-domain is run using OpenFOAM. Due to the fact that, no second order explicit integration scheme are available in openFOAM, some changes had to be made in the proposed coupling method, to pass the fluid resolution from Runge Kutta to backward Euler scheme. Then, the CalculiX and openFOAM adapters as well as the preCICE library have been modified to allow a new coupling scheme, called MCS for monolithic co-simulation, developed in chapter 2. Three dimensional computation of the test cases of the *forward step* and of the *perpendicular flap* were studied. Their results were presented with several configurations, using different time step ratio between the two solvers and using matching or non-matching meshes at the interface. The simulations showed good results with mono time scale and conform meshes. The results of the interface quantities are slightly degraded with the increasing of the time step ratio and with a larger mesh on the solid sub-domain but the general results remains really close to the reference results.

Ross proposed in [101], to define the three challenges of FSI simulation as "discretization heterogeneity", "non matching space and time scales" and "forestalling performance degradation". The coupling method tried to answer to these three challenges. Concerning the forestalling of the performance degradation, thanks to the Schur dual monolithic formulation imposing the normal velocity continuity at the interface, the stability is guaranteed and the method is energy preserving with mono-time scale. Moreover, the discretization methods are heterogeneous since the solid sub-domain is discretized using FEM and implicit integration scheme while the fluid sub-domain is discretized by FVM and explicit integration scheme. Finally, the MCS scheme implemented is multi-time scale, and the preCICE library allows the mapping between non-matching meshes at the fluid-structure interface. In this way, the new proposed method of FSI coupling seems to be well adapted for FSI issues and shows promising results. Naturally, the development of the implementation can be upgraded and further improvements are also considered.

The MCS method is well adapted for FSI simulations and has been validated with benchmarks. It should now be confronted to more complex and realistic FSI simulations. The idea of this work is to propose a general coupling solution for FSI, then, in order to be used for a larger field of simulation, the method should receive some improvement.
First, this work has only presented simulations with linear elastic solid. It would be interesting, in particular in frame of FSI problems, to compute simulations with non-linear solid sub-domain. This can be possible replacing the direct MCS algorithm by an iterative algorithm, with a convergence loop on the macro time step. This could be very costly but can allow the simulation of solid non-linearity in the FSI computation. During the implementation, we have been limited by the OpenFOAM solver. Into the

current version of OpeFOAM, the only solver density based explicit is `rhoCentralFoam`, which is discretized with backward Euler scheme, order one. To increase the order of convergence of the coupling simulation, it would be interesting that all the discretization methods be order two. Recently, a new solver, based on Runge-Kutta integration scheme has been proposed, [**li2200scalability**], but not yet integrated into the openFOAM distribution. This one could be used instead of `rhoCentralFoam` for the MCS software integration, allowing the proposed method from chapter two to be used without any change. Also, the exchanged fields of the current implementation have to be fixed to allow using the mean conservative variables.
Finally, the most important improvement should concern the fluid computation. In this work, the fluid is considered compressible and inviscid. In order to compute more diversified fluid sub-domains, the method should also allow to couple incompressible flows. These kind of fluids are often solved by Navier-Stokes equation with pressure based solver as PISO (Pressure Implicit with Splitting of Operators). As said previously, these solvers sequentially compute the velocity and the pressure, which does not allow the implementation of the MCS coupling method where the direct resolution of the mean conservative variables is required. In [115], the author proposed an implementation of the projection method for pressure–velocity coupling into openFOAM. A solver for incompressible flow, explicit Runge–Kutta based projection was proposed. With some developments, this solver could be used in the frame of MCS implementation, and a larger field of FSI problems could be simulated.

Appendix A

Calculation of the fluid semi-discretized equations

The aim of this appendix is to detail the calculation of governing semi-discretized equations, over the fluid sub-domain, of the proposed method. We start with the weak formulation of the fluid, obtained by the stationarization of the Lagrangian of the fluid-structure problem in monolithic formulation. Thus the objective is to obtain the semi-discretized equations (2.83b). To be lighter, the sub-domain subscripts will be omitted, in this appendix only the fluid sub-domain is treated.

A.1 From weak to strong formulation

The weak form of the fluid sub-domain is made of the terms depending on the virtual field $\delta\underline{\mathbf{X}}_f$ in equation (2.77). There are the stationarization of the variational of the fluid sub-domain, equation (2.80) and the term depending on $\delta\underline{\mathbf{X}}_f$ in the variational of the fluid interface, equation (2.82). The virtual field $\delta\underline{\mathbf{X}}_f$ is re-written $\delta\underline{\mathbf{X}}$ and the weak formulation of the monolithic governing equation over the fluid sub-domain is written as:

$$\int_0^t \int_{\Omega_f} \delta\underline{\mathbf{X}}\big(\frac{\partial \rho\underline{\mathbf{v}}}{\partial t} + \nabla_{\underline{\mathbf{x}}} \cdot (\rho\underline{\mathbf{v}} \otimes \underline{\mathbf{v}} + p\underline{\underline{\mathbf{I}}})\big)\,\mathrm{d}x\mathrm{d}t + \int_0^t \int_{\Gamma_{FS}} \underline{\dot{\Lambda}}(\underline{\mathbf{n}} \cdot \delta\underline{\mathbf{X}})\underline{\mathbf{n}}\,\mathrm{d}X\mathrm{d}t = 0 \quad \text{(A.1)}$$

Let's introduce the continuous function of vertexes selection such as:

$$\underline{\mathbf{l}}(\underline{\mathbf{X}}) = \begin{cases} 0 & \text{if } \underline{\mathbf{X}} \in \Gamma_f, \underline{\mathbf{X}} \notin \Gamma_{FS} \\ \underline{\mathbf{n}} & \text{if } \underline{\mathbf{X}} \in \Gamma_{FS} \end{cases} \quad \text{(A.2)}$$

Then the equation (A.1) is re-written as:

$$\int_0^t \int_{\Omega_f} \delta\underline{\mathbf{X}}\big(\frac{\partial \rho\underline{\mathbf{v}}}{\partial t} + \nabla_{\underline{\mathbf{x}}} \cdot (\rho\underline{\mathbf{v}} \otimes \underline{\mathbf{v}} + p\underline{\underline{\mathbf{I}}})\big)\,\mathrm{d}x\mathrm{d}t + \int_0^t \int_{\Gamma_f} \underline{\mathbf{l}} \cdot \underline{\dot{\Lambda}}(\underline{\mathbf{n}} \cdot \delta\underline{\mathbf{X}})\,\mathrm{d}X\mathrm{d}t = 0 \quad \text{(A.3)}$$

On the boundaries of the sub-domains the ALE coordinate and the Lagrangian coordinate are combined, $\underline{\xi} = \underline{\mathbf{X}}$. Then (A.3) becomes:

$$\int_0^t \int_{\Omega_f} \delta\underline{\mathbf{X}}\big(\frac{\partial \rho\underline{\mathbf{v}}}{\partial t} + \nabla_{\underline{\mathbf{x}}} \cdot (\rho\underline{\mathbf{v}} \otimes \underline{\mathbf{v}} + p\underline{\underline{\mathbf{I}}})\big)\,\mathrm{d}x\mathrm{d}t + \int_0^t \int_{\Gamma_f} \underline{\mathbf{l}} \cdot \underline{\dot{\Lambda}}(\underline{\mathbf{n}} \cdot \delta\underline{\mathbf{X}})\,\mathrm{d}\xi\mathrm{d}t = 0 \quad \text{(A.4)}$$

Applying the divergence theorem, we get:

$$\int_0^t \int_{\Omega_f} \delta\underline{\mathbf{X}}\big(\frac{\partial \rho\underline{\mathbf{v}}}{\partial t} + \nabla_{\underline{\mathbf{x}}} \cdot (\rho\underline{\mathbf{v}} \otimes \underline{\mathbf{v}} + p\underline{\underline{\mathbf{I}}})\big)\,\mathrm{d}x\mathrm{d}t + \int_0^t \int_{\Omega_f} \nabla_{\underline{\xi}} \cdot \underline{\mathbf{l}} \cdot \dot{\underline{\Lambda}}\delta\underline{\mathbf{X}}\,\mathrm{d}\xi\mathrm{d}t = 0 \quad \text{(A.5)}$$

Using the definition of the ALE coordinate, $\mathrm{d}x = \mathbf{J}\mathrm{d}\xi$, equation (A.5) becomes:

$$\int_0^t \int_{\Omega_f} \delta\underline{\mathbf{X}}\big(\frac{\partial \rho\underline{\mathbf{v}}}{\partial t} + \nabla_{\underline{\mathbf{x}}}.(\rho\underline{\mathbf{v}} \otimes \underline{\mathbf{v}} + p\underline{\underline{\mathbf{I}}}) + \mathbf{J}^{-1}\nabla_{\underline{\xi}}.(\underline{\mathbf{l}} \cdot \dot{\underline{\Lambda}})\big)\,\mathrm{d}x\mathrm{d}t = 0 \quad \text{(A.6)}$$

The above equation is true for all $\delta\underline{\mathbf{X}}$ in the virtual space. Thus, the strong formulation in eulerian frame, of the momentum over the fluid domain, for the dual formulation of the fluid-structure problem, is :

$$\frac{\partial \rho\underline{\mathbf{v}}}{\partial t}|_{\underline{\mathbf{x}}} + \nabla_{\underline{\mathbf{x}}}.(\rho\underline{\mathbf{v}} \otimes \underline{\mathbf{v}} + p\underline{\underline{\mathbf{I}}}) + \mathbf{J}^{-1}\nabla_{\underline{\xi}}.(\underline{\mathbf{l}}\dot{\underline{\Lambda}}) = \underline{\mathbf{0}} \qquad (x,t) \in \Omega \times [0,T] \quad \text{(A.7)}$$

Adding the mass conservation equation and the energy conservation equation, equation(A.7) is re-written in vectorial form such as:

$$\frac{\partial \mathbf{U}}{\partial t}|_x + \nabla_x.\mathbf{F} + \mathbf{J}^{-1}\nabla_\xi.(\mathbf{L}\dot{\Lambda}) = \mathbf{0} \qquad (x,t) \in \Omega \times [0,T] \quad \text{(A.8)}$$

From here, the tonsorial notation is leaved and replaced by the bold notation. We define $\mathbf{L} = [\mathbf{0}, l^T, \mathbf{0}]$.

A.2 From eulerian to ALE formulation

We wish now re-write the equation (A.8) in ALE formulation, in the referential of coordinate ξ.
Let defined the temporal derivative at constant ALE coordinate of the conservative variables vector.

$$\frac{\partial \mathbf{U}}{\partial t}|_\xi = \frac{\partial \mathbf{U}}{\partial t}|_x + \nabla_x\mathbf{U}.\frac{\partial x}{\partial t}|_\xi \quad \text{(A.9)}$$

Let's recall the ALE grid velocity, $\mathbf{w} = \frac{\partial x}{\partial t}|_\xi$. Using it the previous equation (A.9) and multiplying this one by $\mathbf{J}$, we get:

$$\mathbf{J}\frac{\partial \mathbf{U}}{\partial t}|_\xi = \mathbf{J}\frac{\partial \mathbf{U}}{\partial t}|_x + \mathbf{J}\nabla_x\mathbf{U}.\mathbf{w} \quad \text{(A.10)}$$

Let's write the jacobian property as following:

$$\frac{\partial \mathbf{J}}{\partial t}|_\xi = \mathbf{J}\nabla_x\mathbf{w} \quad \text{(A.11)}$$

The previous equation (A.11) is multiplied by $\mathbf{U}$:

$$\mathbf{U}\frac{\partial \mathbf{J}}{\partial t}|_\xi = \mathbf{J}\mathbf{U}\nabla_x\mathbf{w} \quad \text{(A.12)}$$

Summing equations (A.10) and (A.10), we get :

$$\mathbf{J}\frac{\partial \mathbf{U}}{\partial t}|_{\xi} + \mathbf{U}\frac{\partial \mathbf{J}}{\partial t}|_{\xi} = \mathbf{J}\frac{\partial \mathbf{U}}{\partial t}|_{x} + \mathbf{J}\nabla_x \mathbf{U}.\mathbf{w} + \mathbf{J}\mathbf{U}\nabla_x \mathbf{w} \tag{A.13}$$

First, the left member is factorized. Then the first term of the right member is replaced by its definition, the equation (A.8). Finally, the last two terms are also factorized. Then equation (A.13) becomes:

$$\frac{\partial \mathbf{J}\mathbf{U}}{\partial t}|_{\xi} = \mathbf{J}\big(-\nabla_x.\mathbf{F} - \frac{1}{\mathbf{J}}\nabla_{\xi}.(\mathbf{L}\dot{\Lambda})\big) + \mathbf{J}\mathbf{U}\nabla_x(\mathbf{U}\mathbf{w}) \tag{A.14}$$

The equation (A.14) is re-organized and the ALE flux is defined as $\tilde{\mathbf{F}} = \mathbf{F} - \mathbf{w}\mathbf{U}$. Thus, the strong formulation on ALE coordinate over the fluid sub-domain of the dual problem is written as:

$$\frac{\partial \mathbf{J}\mathbf{U}}{\partial t}|_{\xi} + \mathbf{J}\nabla_x.\tilde{\mathbf{F}} + \nabla_{\xi}.(\mathbf{L}\dot{\Lambda}) = \mathbf{0} \tag{A.15}$$

A.3 From continuous to integral equations

The objective is now to descretized spatially the equation (A.15). As said previously, the finite volume method is based on the integral formulation, whose the obtaining is developed following.
The equation (A.15) is integrated over a control volume V_i of boundary Γ_i.

$$\int_{V_i} \frac{\partial \mathbf{J}\mathbf{U}}{\partial t}|_{\xi} + \mathbf{J}\nabla_x.\tilde{\mathbf{F}} + \nabla_{\xi}.(\mathbf{L}\dot{\Lambda})\mathrm{d}\xi = \mathbf{0} \tag{A.16}$$

With the ALE formulation, the controlled volume V_i is time independent. Thus, the temporal derivative is moved outside the spatial integral.

$$\frac{d}{dt}\int_{V_i} \mathbf{J}\mathbf{U}\xi\mathrm{d}\xi + \int_{V_i} \mathbf{J}\nabla_x.\tilde{\mathbf{F}}\mathrm{d}\xi + \int_{V_i} \nabla_{\xi}\cdot(\mathbf{L}\dot{\Lambda})\mathrm{d}\xi = \mathbf{0} \tag{A.17}$$

Using the property of the ALE frame, such as $\mathrm{d}\mathbf{x} = \mathbf{J}\mathrm{d}\xi$, equation (A.17) becomes:

$$\frac{d}{dt}\int_{V_i} \mathbf{U}\mathrm{d}\mathbf{x} + \int_{V_i} \nabla_x.\tilde{\mathbf{F}}\mathrm{d}\mathbf{x} + \int_{V_i} \nabla_{\xi}\cdot(\mathbf{L}\dot{\Lambda})\mathrm{d}\xi = \mathbf{0} \tag{A.18}$$

Using the divergence theorem for the last two integral term, we get:

$$\frac{d}{dt}\int_{V_i} \mathbf{U}\mathrm{d}\mathbf{x} + \int_{\Gamma_i} \mathbf{n}.\tilde{\mathbf{F}}\mathrm{d}\mathbf{x} + \int_{\Gamma_i} \mathbf{L}(\mathbf{n}\cdot\dot{\Lambda})\mathrm{d}\xi = \mathbf{0} \tag{A.19}$$

A.4 From integral to semi-discretized equations

Finally the equation (A.19) can be descretized using the finite volumes cell centered method. The conservative variable vector is descretized following equation (2.47) and the numerical flux is discretized by equation (2.62).

The integral of the Lagrange multipliers is discretized using the same approximation as the flux. Nevertheless, the cell face surface s_{if} are also integrated into the discretized interface operator vector. Thus, the continuous Lagrange multipliers vector $\dot{\Lambda}$, which is homogeneous to a pressure appears as the normal forces of interaction discretized $\mathbf{\Lambda}$, homogeneous to a force, in Newton.
Finally, the semi-discretized equations of the fluid sub-domain based on the dual monolithic formulation is written as following:

$$\frac{d}{dt}\left(\Delta V_i \mathbf{U}_i\right) = -\sum_{\forall \Gamma_{ih} \in \Gamma_i} \left(s_{ih}\tilde{\mathbf{F}}_{ih} + \mathbf{L}_{ih}\mathbf{\Lambda}_{ih}\right) \tag{A.20}$$

Appendix B

Lagrange multipliers calculation

The aim of this appendix is to detail the obtaining of the expression (2.94) of the interface operators, from the coupling discretized system of equation (2.84) and the interpolation of solid velocities at the interface, equation (2.85), and Lagrange multipliers, equation (2.86), at the micro time scale. Lets recall of these equations, into the following system:

$$\mathbf{d}_s^m = \mathbf{d}_s^p + \frac{1}{4}\Delta t_s^2 \mathbf{a}_s^m \tag{B.1a}$$

$$\mathbf{v}_s^m = \mathbf{v}_s^p + \frac{1}{2}\Delta t_s \mathbf{a}_s^m \tag{B.1b}$$

$$\tilde{\mathbf{M}}_s \mathbf{a}_s^m = \mathbf{F}_s^m - \mathbf{K}_s \mathbf{d}_s^P + \mathbf{L}_s^T \mathbf{\Lambda}^m \tag{B.1c}$$

$$\Delta V_{f_i}^{j-\frac{1}{2}} \mathbf{U}_{f_i}^{j-\frac{1}{2}} = \Delta V_{f_i}^{j-1} \mathbf{U}_{f_i}^{j-1} - \frac{\Delta t_f}{2} \sum \left(s_{f_{ih}}^{j-1} \tilde{\mathbf{F}}_{f_{ih}}^{j-1} + \mathbf{L}_{f_{ih}}^T \mathbf{\Lambda}_{ih}^{j-\frac{1}{2}} \right) \tag{B.1d}$$

$$\Delta V_{f_i}^{j} \mathbf{U}_{f_i}^{j} = \Delta V_{f_i}^{j-1} \mathbf{U}_{f_i}^{j-1} - \Delta t_f \sum \left(s_{f_{ih}}^{j-\frac{1}{2}} \tilde{\mathbf{F}}_{f_{ih}}^{j-\frac{1}{2}} + \mathbf{L}_{f_{ih}}^T \mathbf{\Lambda}_{ih}^{j} \right) \tag{B.1e}$$

$$\mathbf{L}_s \mathbf{v}_s^{j-\frac{1}{2}} \mathbf{n}_s^{j-\frac{1}{2}} = \mathbf{l}_f \mathbf{v}_{f_{vertex}}^{j-\frac{1}{2}} \mathbf{n}_f^{\frac{1}{2}} \tag{B.1f}$$

$$\mathbf{L}_s \mathbf{v}_s^{j} \mathbf{n}_s^{j} = \mathbf{l}_f \mathbf{v}_{f_{vertex}}^{j} \mathbf{n}_f^{j} \tag{B.1g}$$

$$\mathbf{v}_s^{j-\frac{1}{2}} = (1 - \frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0 + \frac{j-\frac{1}{2}}{m}\mathbf{v}_s^m \tag{B.1h}$$

$$\mathbf{v}_s^{j} = (1 - \frac{j}{m})\mathbf{v}_s^0 + \frac{j}{m}\mathbf{v}_s^m \tag{B.1i}$$

$$\mathbf{\Lambda}^{j-\frac{1}{2}} = (1 - \frac{j-\frac{1}{2}}{m})\mathbf{\Lambda}^0 + \frac{j-\frac{1}{2}}{m}\mathbf{\Lambda}^m \tag{B.1j}$$

$$\mathbf{\Lambda}^{j} = (1 - \frac{j}{m})\mathbf{\Lambda}^0 + \frac{j}{m}\mathbf{\Lambda}^m \tag{B.1k}$$

This system could be solved by the by the Gauss pivot method or the costliest method proposed in section 2.2.2, that came from the following procedure.
First we consider the coupling condition equations at the Runge-Kutta mid step (B.1f) and the micro time step (B.1g):

$$\mathbf{L}_s \mathbf{v}_s^{j-\frac{1}{2}} \mathbf{n}_s^{j-\frac{1}{2}} = \mathbf{l}_f \mathbf{v}_{f_{vertex}}^{j-\frac{1}{2}} \mathbf{n}_f^{j-\frac{1}{2}} \tag{B.2a}$$

$$\mathbf{L}_s \mathbf{v}_s^{j} \mathbf{n}_s^{j} = \mathbf{l}_f \mathbf{v}_{f_{vertex}}^{j} \mathbf{n}_f^{j} \tag{B.2b}$$

Then, the interpolations of the solid velocities at the interface, equation (B.1h) and (B.1i) are injected in (B.2a) and (B.2b) respectively:

$$\mathbf{L}_s\big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0+\frac{j-\frac{1}{2}}{m}\mathbf{v}_s^m\big)\mathbf{n}_s^{j-\frac{1}{2}}=\mathbf{l}_f\mathbf{v}_{f_{vertex}}^{j-\frac{1}{2}}\mathbf{n}_f^{j-\frac{1}{2}} \tag{B.3a}$$

$$\mathbf{L}_s\big((1-\frac{j}{m})\mathbf{v}_s^0+\frac{j}{m}\mathbf{v}_s^m\big)\mathbf{n}_s^{j}=\mathbf{l}_f\mathbf{v}_{f_{vertex}}^{j}\mathbf{n}_f^{j} \tag{B.3b}$$

Then, the fluid velocities at vertices,$\mathbf{v}_{f_{vertex}}$, are re-written according to the extrapolation function called $\mathcal{V}$ defined by the simulation problem, from its velocities values at cell center in the inner fluid sub-domain. In this way, equations (B.3) becomes:

$$\mathbf{L}_s\big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0+\frac{j-\frac{1}{2}}{m}\mathbf{v}_s^m\big)\mathbf{n}_s^{j-\frac{1}{2}}=\mathbf{l}_f\mathcal{V}\big(\mathbf{v}_{f_i}^{j-\frac{1}{2}}\big)\mathbf{n}_f^{j-\frac{1}{2}} \tag{B.4a}$$

$$\mathbf{L}_s\big((1-\frac{j}{m})\mathbf{v}_s^0+\frac{j}{m}\mathbf{v}_s^m\big)\mathbf{n}_s^{j}=\mathbf{l}_f\mathcal{V}\big(\mathbf{v}_{f_i}^{j}\big)\mathbf{n}_f^{j} \tag{B.4b}$$

The next step is to replace the solid velocity value at t^m with equation (B.1b) :

$$\begin{aligned}\mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0+\frac{j-\frac{1}{2}}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\mathbf{a}_s^m\big)\Big)\mathbf{n}_s^{j-\frac{1}{2}}&\\ =\mathbf{l}_f\mathcal{V}\big(\mathbf{v}_{f_i}^{j-\frac{1}{2}}\big)\mathbf{n}_f^{j-\frac{1}{2}}&\end{aligned} \tag{B.5a}$$

$$\mathbf{L}_s\Big((1-\frac{j}{m})\mathbf{v}_s^0+\frac{j}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\mathbf{a}_s^m\big)\Big)\mathbf{n}_s^{j}=\mathbf{l}_f\mathcal{V}\big(\mathbf{v}_{f_i}^{j}\big)\mathbf{n}_f^{j} \tag{B.5b}$$

One can now replace the fluid velocities by their values. These velocities are expressed according to the fluid mean conservatives variables, using the equation (2.92):

$$\begin{aligned}\mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0+\frac{j-\frac{1}{2}}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\mathbf{a}_s^m\big)\Big)\mathbf{n}_s^{j-\frac{1}{2}}&\\ =\mathbf{l}_f\mathcal{V}\Big(\frac{\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j-\frac{1}{2}}}{\big(\Delta V\rho\big)_{f_i}^{j-\frac{1}{2}}}\Big)\mathbf{n}_f^{j-\frac{1}{2}}&\end{aligned} \tag{B.6a}$$

$$\begin{aligned}\mathbf{L}_s\Big((1-\frac{j}{m})\mathbf{v}_s^0+\frac{j}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\mathbf{a}_s^m\big)\Big)\mathbf{n}_s^{j}&\\ =\mathbf{l}_f\mathcal{V}\Big(\frac{\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j}}{\big(\Delta V\rho\big)_{f_i}^{j}}\Big)\mathbf{n}_f^{j}&\end{aligned} \tag{B.6b}$$

Finally, the the Lagrange multipliers vector appears replacing the solid acceleration and $\mathbf{s}_s^m$ by their values equation (B.1c), and the fluid mean momentum $\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j-\frac{1}{2}}$ and $\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j}$ using equations (B.1d) and (B.1e):

$$\mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0+\frac{j-\frac{1}{2}}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\big(\tilde{\mathbf{M}}_s^{-1}(\mathbf{F}_s^m-\mathbf{K}_s\mathbf{d}_s^P+\mathbf{L}_s^T\Lambda^m)\big)\big)\Big)\mathbf{n}_s^{j-\frac{1}{2}}$$
$$=\mathbf{l}_f\mathcal{V}\Big(\frac{1}{\big(\Delta V\rho\big)_{f_i}^{j-\frac{1}{2}}}\big(\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j-1}-\frac{\Delta t_f}{2}\sum\big(s_{f_{ih}}^{j-1}\tilde{\mathbf{F}}_{f_{ih}}^{j-1}+\mathbf{l}_{f_{ih}}^T\Lambda_{ih}^{j-\frac{1}{2}}\big)\big)\Big)\mathbf{n}_f^{j-\frac{1}{2}}\quad\text{(B.7a)}$$

$$\mathbf{L}_s\Big((1-\frac{j}{m})\mathbf{v}_s^0+\frac{j}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\big(\tilde{\mathbf{M}}_s^{-1}(\mathbf{F}_s^m-\mathbf{K}_s\mathbf{d}_s^P+\mathbf{L}_s^T\Lambda^m)\big)\big)\Big)\mathbf{n}_s^{j}$$
$$=\mathbf{l}_f\mathcal{V}\Big(\frac{1}{\big(\Delta V\rho\big)_{f_i}^{j}}\big(\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j-1}-\Delta t_f\sum\big(s_{f_{ih}}^{j-\frac{1}{2}}\tilde{\mathbf{F}}_{f_{ih}}^{j-\frac{1}{2}}+\mathbf{l}_{f_{ih}}^T\Lambda_{ih}^{j}\big)\big)\Big)\mathbf{n}_f^{j-\frac{1}{2}}\quad\text{(B.7b)}$$

The last step before simplification, is to use the equation (B.1j) into (B.7a), respectively (B.1k) into (B.7b) to replace the Lagrange multipliers at the macro time step.

$$\mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}_s^0+\frac{j-\frac{1}{2}}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\big(\tilde{\mathbf{M}}_s^{-1}(\mathbf{F}_s^m-\mathbf{K}_s\mathbf{d}_s^P$$
$$+\mathbf{L}_s^T(\frac{m}{j-\frac{1}{2}}\Lambda^j-(\frac{m}{j-\frac{1}{2}}-1)\mathbf{\Lambda}^0)\big)\big)\Big)\mathbf{n}_s^{j-\frac{1}{2}}$$
$$=\mathbf{l}_f\mathcal{V}\Big(\frac{1}{\big(\Delta V\rho\big)_{f_i}^{j-\frac{1}{2}}}\big(\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j-1}-\frac{\Delta t_f}{2}\sum\big(s_{f_{ih}}^{j-1}\tilde{\mathbf{F}}_{f_{ih}}^{j-1}+\mathbf{l}_{f_{ih}}^T\Lambda_{ih}^{j-\frac{1}{2}}\big)\big)\Big)\mathbf{n}_f^{j-\frac{1}{2}}\quad\text{(B.8a)}$$

$$\mathbf{L}_s\Big((1-\frac{j}{m})\mathbf{v}_s^0+\frac{j}{m}\big(\mathbf{v}_s^p+\frac{1}{2}\Delta t_s\big(\tilde{\mathbf{M}}_s^{-1}(\mathbf{F}_s^m-\mathbf{K}_s\mathbf{d}_s^P$$
$$+\mathbf{L}_s^T(\frac{m}{j}\Lambda^j-(\frac{m}{j}-1)\mathbf{\Lambda}^0)\big)\big)\Big)\mathbf{n}_s^{j}$$
$$=\mathbf{l}_f\mathcal{V}\Big(\frac{1}{\big(\Delta V\rho\big)_{f_i}^{j}}\big(\big(\Delta V\rho\mathbf{v}\big)_{f_i}^{j-1}-\Delta t_f\sum\big(s_{f_{ih}}^{j-\frac{1}{2}}\tilde{\mathbf{F}}_{f_{ih}}^{j-\frac{1}{2}}+\mathbf{l}_{f_{ih}}^T\Lambda_{ih}^{j}\big)\big)\Big)\mathbf{n}_f^{j-\frac{1}{2}}\quad\text{(B.8b)}$$

Then, into solid terms of each equation(the first member) the terms depending on $\mathbf{\Lambda}^0$ can be simplified. Then the equations are reorganized; the terms depending on the current forces of interaction, $\mathbf{\Lambda}^{j-\frac{1}{2}}$ and $\mathbf{\Lambda}^j$ are put on the left member of equation (B.8a) and (B.8b) respectively, while the term non depending on the current forces of interaction are put into the right side.

$$\Big(\frac{1}{2}\Delta t_s \mathbf{L}_s \tilde{\mathbf{M}}_s^{-1}\mathbf{L}_s^T + \frac{1}{2}\Delta t_f \mathbf{l}_f \mathbf{M}_f^{j-\frac{1}{2}^{-1}}\mathbf{l}_f^T\Big)\mathbf{\Lambda}^{j-\frac{1}{2}} =$$
$$\mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\big(\mathbf{v}_s^0 + \frac{1}{2}\Delta t_s \tilde{\mathbf{M}}_s^{-1}\mathbf{L}_s^T\mathbf{\Lambda}^0\big) + \frac{j-\frac{1}{2}}{m}\big(\mathbf{v}_s^p + \frac{1}{2}\Delta t_s\big(\tilde{\mathbf{M}}_s^{-1}(\mathbf{F}_s^m - \mathbf{K}_s\mathbf{d}_s^P)\big)\big)\Big)\mathbf{n}_s^{j-\frac{1}{2}}$$
$$+\mathbf{l}_f\mathcal{V}\Big(\frac{1}{\big(\Delta V\rho\big)_{f_i}^{j-\frac{1}{2}}}\big((\Delta V\rho\mathbf{v})_{f_i}^{j-1} - \frac{\Delta t_f}{2}\sum\big(s_{f_{ih}}^{j-1}\tilde{\mathbf{F}}_{f_{ih}}^{j-1}\big)\big)\Big)\mathbf{n}_f^{j-\frac{1}{2}}$$
$$\Big(\frac{1}{2}\Delta t_s \mathbf{L}_s \tilde{\mathbf{M}}_s^{-1}\mathbf{L}_s^T + \Delta t_f \mathbf{l}_f \mathbf{M}_f^{j^{-1}}\mathbf{l}_f^T\Big)\mathbf{\Lambda}^{j} =$$
$$\mathbf{L}_s\Big((1-\frac{j}{m})\big(\mathbf{v}_s^0 + \frac{1}{2}\Delta t_s \tilde{\mathbf{M}}_s^{-1}\mathbf{L}_s^T\mathbf{\Lambda}^0\big) + \frac{j}{m}\big(\mathbf{v}_s^p + \frac{1}{2}\Delta t_s\big(\tilde{\mathbf{M}}_s^{-1}(\mathbf{F}_s^m - \mathbf{K}_s\mathbf{d}_s^P)\big)\big)\Big)\mathbf{n}_s^{j}$$
$$+\mathbf{l}_f\mathcal{V}\Big(\frac{1}{\big(\Delta V\rho\big)_{f_i}^{j}}\big((\Delta V\rho\mathbf{v})_{f_i}^{j-1} - \Delta t_f\sum\big(s_{f_{ih}}^{j-\frac{1}{2}}\tilde{\mathbf{F}}_{f_{ih}}^{j-\frac{1}{2}}\big)\big)\Big)\mathbf{n}_f^{j}$$

(B.9a)

(B.9b)

Where, the time dependent fluid mass matrix, $\mathbf{M}_f^j$, is defined as a diagonal matrix such as: $\mathbf{M}_{f_i}^j = \mathcal{V}\Big(\big(\Delta V\rho\big)_{f_i}^j\Big)$

Moreover, in the first member, the Steklov-Poincare operators $\mathbf{H}_f^j$ respectively $\mathbf{H}_f^{j-\frac{1}{2}}$ and $\mathbf{H}_s$ appear, as defined by equations (2.95).
Finally, the *free-link* decomposition appears, (2.7). In this way, using the fluid *free* mean momentum, equation (2.90), and the *free* solid velocities (2.88b) and the *link* solid velocities (2.89b), the equation (B.9) are re-written such as :

$$\Big(\mathbf{H}_s + \mathbf{H}_f^{j-\frac{1}{2}}\Big)\mathbf{\Lambda}^{j-\frac{1}{2}} = \mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\big(\mathbf{v}_s^0 - \mathbf{v}_{s_{link}}^0\big) + \frac{j-\frac{1}{2}}{m}\mathbf{v}_{s_{free}}^m\Big)\mathbf{n}_s^{j-\frac{1}{2}}$$
$$+\mathbf{l}_f\mathcal{V}\Big(\frac{\big(\Delta V\rho\mathbf{v}\big)_{f_{free_i}}^{j-\frac{1}{2}}}{\big(\Delta V\rho\big)_{f_i}^{j-\frac{1}{2}}}\Big)\mathbf{n}_f^{j-\frac{1}{2}} \quad \text{(B.10a)}$$
$$\Big(\mathbf{H}_s + \mathbf{H}_f^{j}\Big)\mathbf{\Lambda}^{j} = \mathbf{L}_s\Big((1-\frac{j}{m})\big(\mathbf{v}_s^0 - \mathbf{v}_{s_{link}}^0\big) + \frac{j}{m}\mathbf{v}_{s_{free}}^m\Big)\mathbf{n}_s^{j}$$
$$+\mathbf{l}_f\mathcal{V}\Big(\frac{\big(\Delta V\rho\mathbf{v}\big)_{f_{free_i}}^{j}}{\big(\Delta V\rho\big)_{f_i}^{j}}\Big)\mathbf{n}_f^{j} \quad \text{(B.10b)}$$

Then using then the equation (2.87) on the solid velocities at t^0, and simplifying the expression of fluid velocities the equations (B.12) become:

$$\Big(\mathbf{H}_s + \mathbf{H}_f^{j-\frac{1}{2}}\Big)\mathbf{\Lambda}^{j-\frac{1}{2}} = \mathbf{L}_s\Big((1-\frac{j-\frac{1}{2}}{m})\mathbf{v}^0_{s_{free}} + \frac{j-\frac{1}{2}}{m}\mathbf{v}^m_{s_{free}}\Big)\mathbf{n}_s^{j-\frac{1}{2}} + \mathbf{l}_f\mathcal{V}\Big(\mathbf{v}^{j-\frac{1}{2}}_{f_{free_i}}\Big)\mathbf{n}_f^{j-\frac{1}{2}}$$

$$\Big(\mathbf{H}_s + \mathbf{H}_f^{j}\Big)\mathbf{\Lambda}^{j} = \mathbf{L}_s\Big((1-\frac{j}{m})\mathbf{v}^0_{s_{free}} + \frac{j}{m}\mathbf{v}^m_{s_{free}}\Big)\mathbf{n}_s^{j} + \mathbf{l}_f\mathcal{V}\Big(\mathbf{v}^{j}_{f_{free_i}}\Big)\mathbf{n}_f^{j}$$

(B.11a)

(B.11b)

Then, using the solid velocities interpolation (B.1h) and (B.1i) and the definition of the function $\mathcal{V}$, the found equations (2.94) appear :

$$\Big(\mathbf{H}_s + \mathbf{H}_f^{j-\frac{1}{2}}\Big)\mathbf{\Lambda}^{j-\frac{1}{2}} = \mathbf{L}_s\mathbf{v}^{j-\frac{1}{2}}_{s_{free}}\mathbf{n}_s^{j-\frac{1}{2}} + \mathbf{l}_f\mathbf{l}_f\mathbf{v}^{j-\frac{1}{2}}_{vertex_{f_{free}}}\mathbf{n}_f^{j-\frac{1}{2}} \tag{B.12a}$$

$$\Big(\mathbf{H}_s + \mathbf{H}_f^{j}\Big)\mathbf{\Lambda}^{j} = \mathbf{L}_s\mathbf{v}^{j}_{s_{free}}\mathbf{n}_s^{j} + \mathbf{l}_f\mathbf{v}^{j}_{vertex_{f_{free}}}\mathbf{n}_f^{j} \tag{B.12b}$$

Appendix C

Implementation into CalculiX and OpenfOAM adapters

C.1 XML configuration file for the forward step problem

C.2 OpenFOAM adapter: Velocity.C

```
#include "Velocity.H"
using namespace Foam;

preciceAdapter::FSI::Velocity::Velocity
(
        const Foam::fvMesh& mesh
)
:
mesh_(mesh),
U_(
        const_cast<volVectorField*>
        (
                &mesh.lookupObject<volVectorField>("U")
        )
)
{
    dataType_ = vector;
}

void preciceAdapter::FSI::Velocity::write(double * buffer, bool meshConnectivity,
                                          const unsigned int dim)
{
    int bufferIndex = 0;

    // For every boundary patch of the interface
    for (uint j = 0; j < patchIDs_.size(); j++)
    {
        int patchID = patchIDs_.at(j);
        //- set-up interpolator
        primitivePatchInterpolation patchInterpolator
                (
```

```
                        U_->mesh().boundaryMesh()[patchID]
                );

        vectorField n = mesh_.boundary()[patchID].nf();
        vectorField UFaceValues = U_->boundaryFieldRef()[patchID];

        //- Perform interpolation of normal vector
        vectorField UPointValues
                    =
                    patchInterpolator.faceToPointInterpolate((n&UFaceValues)*n);

        // For every cell of the patch
        forAll(UPointValues, i)
        {
            // Copy the normal free velocities into the buffer
            // x-dimension
            buffer[bufferIndex++] = UPointValues[i].x();

            // y-dimension
            buffer[bufferIndex++] = UPointValues[i].y();

            if(dim == 3)
                // z-dimension
                buffer[bufferIndex++] = UPointValues[i].z();
        }
    }
}
```

C.3 OpenFOAM adapter: Mass.C

```
#include "Mass.H"
using namespace Foam;

preciceAdapter::FSI::Mass::Mass
(
        const Foam::fvMesh& mesh
)
:
mesh_(mesh)
{
    dataType_ = scalar;
}

void preciceAdapter::FSI::Mass::write(double * buffer, bool meshConnectivity,
                                      const unsigned int dim)
{
    volScalarField rho_ = mesh_.lookupObject<volScalarField>("rho");

    volScalarField cellMass
            (
```

```
                    IOobject
                            (
                                    "cellMass",
                                    mesh_.time().timeName(),
                                    mesh_,
                                    IOobject::NO_READ,
                                    IOobject::AUTO_WRITE
                            ),
                    mesh_,
                    dimensionedScalar("zero", dimMass, 0.0)
            );
    cellMass.ref()= mesh_.V()*rho_();
    cellMass.write();

    int bufferIndex = 0;

    // For every boundary patch of the interface
    for (uint j = 0; j < patchIDs_.size(); j++)
    {
        int patchID = patchIDs_.at(j);

        scalarField massFaceValues = cellMass.boundaryFieldRef()[patchID];

        // For every feces of the patch
        forAll(massFaceValues, i)
        {
            // Copy the mass into the buffer
            buffer[bufferIndex++] = massFaceValues[i];
            }
        }
}
```

C.4 OpenFOAM adapter: VelocityLink.C

```
#include "VelocityLink.H"
#include "TimeState.H"
#include "surfaceInterpolate.H"
using namespace Foam;

preciceAdapter::FSI::VelocityLink::VelocityLink
        (
                const Foam::fvMesh& mesh,
                const Foam::Time& runTime
        )
        :
        U_(
                const_cast<volVectorField*>
                (
                        &mesh.lookupObject<volVectorField>("U")
                )
```

```
        ),
        mesh_(mesh),
        runTime_(runTime)
{
    dataType_ = vector;
}

void preciceAdapter::FSI::VelocityLink::read(double * buffer,
                                             const unsigned int dim)
{
    int bufferIndex = 0; // buffer get lagrange multipliers
    volScalarField rho_ = mesh_.lookupObject<volScalarField>("rho");

    // For every boundary patch of the interface
    for (uint j = 0; j < patchIDs_.size(); j++) {
        int patchID = patchIDs_.at(j);

        // For every faces of the patch
        forAll(mesh_.boundaryMesh()[patchID], faceI)
        {
            //cell number whose contain the current face
            const label& ownerI = mesh_.boundaryMesh()[patchID].faceCells()[faceI];
            //link velocity correction at cell center
            U_->ref()[ownerI].x()
                =
                U_->boundaryFieldRef()[patchID][faceI].x() -
                runTime_.deltaT().value()/(mesh_.V()[ownerI]*rho_.ref()[ownerI])
                *buffer[bufferIndex++];
            U_->ref()[ownerI].y()
                =
                U_->boundaryFieldRef()[patchID][faceI].y()
                runTime_.deltaT().value()/(mesh_.V()[ownerI]*rho_.ref()[ownerI])
                *buffer[bufferIndex++];
            U_->ref()[ownerI].z()
                =
                U_->boundaryFieldRef()[patchID][faceI].z() -
                runTime_.deltaT().value()/(mesh_.V()[ownerI]
                *rho_.ref()[ownerI])*buffer[bufferIndex++];
            //correct boundary values for next flux computation
            U_->boundaryFieldRef()[patchID][0].x() = U_->ref()[ownerI].x();
            U_->boundaryFieldRef()[patchID][0].y() = U_->ref()[ownerI].y();
            U_->boundaryFieldRef()[patchID][0].z() = U_->ref()[ownerI].z();
        }
        // writhe total state
        U_->write();
    }
}
```

C.5 OpenFOAM adapter: GridDisp.C

```
#include "GridDisp.H"
#include "TimeState.H"
#include "volPointInterpolation.H"
using namespace Foam;

preciceAdapter::FSI::GridDisp::GridDisp
        (
                const Foam::fvMesh& mesh,
                const Foam::Time& runTime
        )
        :
        pointDisplacement_(
                const_cast<pointVectorField*>
                (
                        &mesh.lookupObject<pointVectorField>("pointDisplacement")
                )
        ),
        U_(
                const_cast<volVectorField*>
                (
                        &mesh.lookupObject<volVectorField>("U")
                )
        ),
        mesh_(mesh),
        runTime_(runTime)
{
    dataType_ = vector;
}

void preciceAdapter::FSI::GridDisp::read(double * buffer, const unsigned int dim)
{
    int bufferIndex = 0; //buffer get tangential normal velocity

    // For every boundary patch of the interface
    for (uint j = 0; j < patchIDs_.size(); j++) {
        int patchID = patchIDs_.at(j);
        // Get the displacement on the patch
        fixedValuePointPatchVectorField &pointDisplacementFluidPatch
                (
                    refCast<fixedValuePointPatchVectorField>
                            (
                                pointDisplacement_->boundaryFieldRef()[patchID]
                            )
                );
        //- set-up interpolator
        primitivePatchInterpolation patchInterpolator
                (
                        U_->mesh().boundaryMesh()[patchID]
                );
```

```
        // Get normal faces of the interface
        vectorField n = mesh_.boundary()[patchID].nf();
        vectorField UFaceValues = U_->boundaryFieldRef()[patchID];

        //- Perform interpolation of normal vector
        vectorField UPointValues
                    =
                    patchInterpolator.faceToPointInterpolate((n&UFaceValues)*n);

        // For every point of the patch
        forAll(pointDisplacement_->boundaryFieldRef()[patchID], i)
        {
            // Set the nodal displacement as explicit prediction for next time step
            pointDisplacementFluidPatch[i][0]
                =
                pointDisplacementFluidPatch[i][0] + runTime_.deltaT().value()
                *(UPointValues[i].x() + buffer[bufferIndex++]);
            pointDisplacementFluidPatch[i][1]
                =
                pointDisplacementFluidPatch[i][1] + runTime_.deltaT().value()
                *(UPointValues[i].y() +buffer[bufferIndex++]);
            pointDisplacementFluidPatch[i][2]
                =
                pointDisplacementFluidPatch[i][2]+runTime_.deltaT().value()
                *(UPointValues[i].z()+buffer[bufferIndex++]);

        }
        //correction boundary bit no writing, correction for next time step
        pointDisplacement_->correctBoundaryConditions();
    }
}
```

C.6 CalculiX adapter: WriteCouplingData

```
#include <stdlib.h>
#include "PreciceInterface.h"
#include "ConfigReader.h"
#include "precice/SolverInterfaceC.h"

void Precice_WriteCouplingData( SimulationData * sim )
{

  printf( "---[preciceAdapter] Adapter writing coupling data...\n" );
  fflush( stdout );

  PreciceInterface ** interfaces = sim->preciceInterfaces;
  int numInterfaces = sim->numPreciceInterfaces;
  int i, j;
  int iset;
```

```
if( precicec_isWriteDataRequired( sim->solver_dt )
      || precicec_isActionRequired( "write-initial-data" ) )
{
  for( i = 0 ; i < numInterfaces ; i++ )
  {
    for( j = 0 ; j < interfaces[i]->numWriteData ; j++ )
    {
      switch( interfaces[i]->writeData[j] )
      {
        case TEMPERATURE:
          ...
          break;
        case HEAT_FLUX:
          ...
          break;
        case CONVECTION:
          ...
          break;
        case DISPLACEMENTS:
          ...
          break;
        case DISPLACEMENTDELTAS:
          ...
          break;
        case NVELOCITIES:
          iset = interfaces[i+2]->faceSetID + 1; // TODO: define correctly +2
          FORTRAN( getnormalvel, (sim->co,
                                  sim->mi,
                                  &iset,
                                  sim->istartset,
                                  sim->iendset,
                                  sim->ipkon,
                                  *sim->lakon,
                                  sim->kon,
                                  sim->ialset,
                                  sim->veold,
                                  interfaces[i]->nodeVectorData
                                )
                   );
          precicec_writeBlockVectorData(interfaces[i]->velocitiesDataID,
                  interfaces[i]->numNodes,interfaces[i]->preciceNodeIDs,
                  interfaces[i]->nodeVectorData );
          break;
        case TVELOCITIES:
          iset = interfaces[i]->faceSetID + 1; //Adjust index before calling Fortran
          FORTRAN( gettangentialvel, (sim->co,
                                      sim->mi,
                                      &iset,
                                      sim->istartset,
                                      sim->iendset,
```

```
                                        sim->ipkon,
                                        *sim->lakon,
                                        sim->kon,
                                        sim->ialset,
                                        sim->veold,
                                        interfaces[i]->nodeVectorData
                                    )
                            );
              precicec_writeBlockVectorData( interfaces[i]->velocitiesDataID,
                  interfaces[i]->numNodes, interfaces[i]->preciceNodeIDs,
                  interfaces[i]->nodeVectorData );
              break;
            case POSITIONS:
                            ...
                            break;
            case FORCES:
              getNodeForces( interfaces[i]->nodeIDs, interfaces[i]->numNodes,
               interfaces[i]->dim, sim->fn, sim->mt, interfaces[i]->nodeVectorData );
              precicec_writeBlockVectorData( interfaces[i]->forcesDataID,
               interfaces[i]->numNodes, interfaces[i]->preciceNodeIDs,
               interfaces[i]->nodeVectorData );
                            break;
                }
            }
          }
          if( precicec_isActionRequired( "write-initial-data" ) )
          {
                  precicec_markActionFulfilled( "write-initial-data" );
          }
    }
}
```

C.7 CalculiX adapter: Link computation

```
/*If the coupling method is MCS, the solid state is corrected (=free+link)*/
if(Precice_isMcsCouplingScheme()){
    /*clear all the external contributions*/
    for(k=0;k<*nforc;++k){
        xforc[k] = 0 ;
    }
    for(k=0;k<*nload;++k){
        xload[k] = 0 ;
    }

    /*Read the interacton force into xforc*/
    Precice_ReadCouplingData( &simulationData );

    /*Constrct the fext vector (only Lambda for MCS correction)*/
    FORTRAN(mafillsmforc,(nforc,ndirforc,nodeforc,xforc,nactdof,
            fext,ipompc,nodempc,coefmpc,mi,&rhsi,fnext,
```

```
            nmethod,ntrans,inotr,trab,co));

    /*Computation of the second memeber b=fext
          (f internal contribution have ever been take
                    into account into the free computation)*/
    for(k=0;k<neq[0];++k){
        b[k]=fext[k];
    }

    /* compute acc, b<-Mtilde*b */
    if(*isolver==0){
        ...
    }


    /*Compute the total state = free+link for displacement, velocity and acceleration*/
    results(co,nk,kon,ipkon,lakon,ne,v,stn,inum,stx,
            elcon,nelcon,rhcon,nrhcon,alcon,nalcon,alzero,ielmat,
            ielorien,norien,orab,ntmat_,t0,t1act,ithermal,
            prestr,iprestr,filab,eme,emn,een,iperturb,
            f,fn,nactdof,&iout,qa,vold,b,nodeboun,
            ndirboun,xbounact,nboun,ipompc,
            nodempc,coefmpc,labmpc,nmpc,nmethod,cam,&neq[1],veold,accold,
            &bet,&gam,&dtime,&time,ttime,plicon,nplicon,plkcon,nplkcon,
            xstateini,xstiff,xstate,npmat_,epn,matname,mi,&ielas,&icmd,
            ncmat_,nstate_,stiini,vini,ikboun,ilboun,ener,enern,emeini,
            xstaten,eei,enerini,cocon,ncocon,set,nset,istartset,iendset,
            ialset,nprint,prlab,prset,qfx,qfn,trab,inotr,ntrans,fmpc,
            nelemload,nload,ikmpc,ilmpc,istep,&iinc,springarea,
            &reltime,&ne0,thicke,shcon,nshcon,
            sideload,xloadact,xloadold,&icfd,inomat,pslavsurf,pmastsurf,
            mortar,islavact,cdn,islavnode,nslavnode,ntie,clearini,
            islavsurf,ielprop,prop,energyini,energy,&kscale,iponoel,
            inoel,nener,orname,network,ipobody,xbodyact,ibody,typeboun);


    /*clear all the interface contributions for the next time step free computation*/
    for(k=0;k<*nforc;++k){
        xforc[k] = 0 ;
    }
}
```

C.8 Action: computeHf.py

```
def performAction(time, dt, sourceData, targetData):

    s= '---[precice] Python action Compute Hf'
    print(s, flush=True)

    Mf = sourceData # store (reference to) sourceData for later use
```

```
    Hf = dt*np.linalg.inv(np.dot(Mf,np.eye(Mf.shape[0]*3))
    np.save('Hf.npy', Hf)
```

C.9 Action: computeLambda.py

```
def performAction(time, dt, sourceData, targetData):

    s= '---[precice] Python Action Compute Lambda'
    print(s, flush=True)

    m = 1  #Time step ratio, ToDo reat it from xml config
    dt_f = dt/m

    #Computation of the structural operator at initialization
    if (0<time and time <= dt) :
        #Read and create Mtilde
        diag_file = open('Mtilde_diag.txt')
        uptri_file = open('Mtilde_tri.txt')
        diag_lines = diag_file.readlines()
        uptri_lines = uptri_file.readlines()
        Mtilde = np.zeros((len(diag_lines),len(diag_lines)))

        for i in range(len(diag_lines)) :
            Mtilde[i,i] = float(diag_lines[i])
        k = 0
        for i in range(1,len(diag_lines)-1) :
            for j in range(i):
                Mtilde[i,j] = float(uptri_lines[k])
                Mtilde[j,i] = float(uptri_lines[k])
                k = k+1

        #Steklov Pointcare operator
        Hs = 0.5*dt*np.linalg.inv(Mtilde)
        np.save('Hs.npy', Hs)

        #Save first free old vel in case of multi time step interpolation required
        vel_free_solid_old = np.zeros(Hs.shape[0])
        np.save('vel_free_solid_old.npy', vel_free_solid_old)

    if time >= dt :
        j = int(time/dt_f)%m
        if j == 0:
            j=m

        #Read the free velocity
        vel_free_f = targetData #free normal velocities fluid at micro time step
        vel_free_s= sourceData #free normal velocities solid at macro time step
        #solid micro time step interpolation
        vel_free_s_old= np.load('vel_free_solid_old.npy')
        vel_free_s_j = (1-j/m)*vel_free_s_old + j/m*vel_free_s
```

```
        #Read operator
        H_s = np.load('Hs.npy')
        H_f = np.load('Hf.npy')

        #Compute interaction force
        Lambda = np.dot(np.linalg.inv(H_s+H_f), vel_free_f-vel_free_s)
        for i in range(targetData.shape[0]):
            myTargetData[i] = Lambda[i]
        np.save('Lambda.npy', myTargetData)

        #save old solid for next micro time interpolation
        if j ==m:
            np.save('vel_free_solid_old.npy', vel_free_s)
```

C.10 Action: transferLambda.py

```
def performAction(time, dt, sourceData, targetData):

    s= '---[precice] Python Action Transfer Lambda'
    print(s, flush=True)

    if time != 0. :
        Lambda = np.load('Lambda.npy')
        for i in range(targetData.shape[0]):
            targetData[i] = Lambda[i]
```

C.11 Action: computeMicro.py

```
def performAction(time, dt, sourceData, targetData):

    s= '---[precice] Python Action Compute micro'
    print(s, flush=True)

    m = 1  #Time step ratio, ToDo reat it from xml config

    if (m !=1) : #if multi time step
        #Saved init old value
        if (0<time and time <= dt) :
            vel_tan_old = np.zeros(sourceData)
            np.save('vel_tan_old.npy', vel_tan_old)

        if time >= dt :
            j = int(time/dt_f)\%m
            if j == 0:
                j=m
            #tangential velocities solid at macro time step
            vel_tan_m= sourceData
            #tangential velocities solid at  previous macro time step
```

```
vel_tan_old= np.load('vel_tan_old.npy')
#tangential velocities at previous micro time step
vel_tan_j1 = (1-(j-1)/m)*vel_tan_old + (j-1)/m*vel_tan_m

for i in range(targetData.shape[0]):
    myTargetData[i] = vel_tan_j1[i]

#save old solid for next micro time interpolation
if j ==m:
    np.save('vel_tan_old.npy', vel_tan_m)
```

Bibliography

[1] J Achenbach. *Wave propagation in elastic solids*. Vol. 16. Elsevier, 2012.

[2] M. Alimohammadi, J. M. Sherwood, M. Karimpour, O. Agu, S. Balabani, and V. Díaz-Zuccarini. "Aortic dissection simulation models for clinical support: fluid-structure interaction vs. rigid wall models". In: *Biomedical engineering online* 14.1 (2015), pp. 1–16.

[3] J. Alonso and A. Jameson. "Fully-implicit time-marching aeroelastic solutions". In: *32nd Aerospace Sciences Meeting and Exhibit*. 1994, p. 56.

[4] A. M. Amini, D. Dureisseix, and P. Cartraud. "Multi-scale domain decomposition method for large-scale structural analysis with a zooming technique: Application to plate assembly". In: *International Journal for Numerical Methods in Engineering* 79.4 (2009), pp. 417–443.

[5] O. Andrianarison and R. Ohayon. "Compressibility and gravity effects in internal fluid–structure vibrations: Basic equations and appropriate variational formulations". In: *Computer Methods in Applied Mechanics and Engineering* 195.17-18 (2006), pp. 1958–1972.

[6] ANSYS. *ANSYS Mechanical User's Guide*. 2022.

[7] ANSYS. *Fluent User's Guide*. 2022.

[8] F. P. Baaijens. "A fictitious domain/mortar element method for fluid–structure interaction". In: *International Journal for Numerical Methods in Fluids* 35.7 (2001), pp. 743–761.

[9] K.-J. Bathe, H. Zhang, and S. Ji. "Finite element analysis of fluid flows fully coupled with structural interactions". In: *Computers & structures* 72.1-3 (1999).

[10] Y. Bazilevs, M.-C. Hsu, J. Kiendl, R. Wüchner, and K.-U. Bletzinger. "3D simulation of wind turbine rotors at full scale. Part II: Fluid–structure interaction modeling with composite blades". In: *International Journal for Numerical Methods in Fluids* 65.1-3 (2011).

[11] Y. Bazilevs, V. M. Calo, Y. Zhang, and T. J. Hughes. "Isogeometric fluid–structure interaction analysis with applications to arterial blood flow". In: *Computational Mechanics* 38.4 (2006), pp. 310–322.

[12] A. Beckert and H. Wendland. “Multivariate interpolation for fluid-structure-interaction problems using radial basis functions”. In: *Aerospace Science and Technology* 5.2 (2001), pp. 125–134.

[13] F.-K. Benra, H. J. Dohmen, J. Pei, S. Schuster, and B. Wan. “A comparison of one-way and two-way coupling methods for numerical analysis of fluid-structure interactions”. In: *Journal of applied mathematics* (2011).

[14] G. Bertaglia, V. Caleffi, L. Pareschi, and A. Valiani. “Uncertainty quantification of viscoelastic parameters in arterial hemodynamics with the a-FSI blood flow model”. In: *Journal of Computational Physics* 430 (2021), p. 110102.

[15] M. Bhardwaj, D. Day, C. Farhat, M. Lesoinne, K. Pierson, and D. Rixen. “Application of the FETI method to ASCI problems—scalability results on 1000 processors and discussion of highly heterogeneous problems”. In: *International Journal for numerical methods in engineering* 47.1-3 (2000), pp. 513–535.

[16] F. J. Blom. “A monolithical fluid-structure interaction algorithm applied to the piston problem”. In: *Computer methods in applied mechanics and engineering* 167.3-4 (1998).

[17] A. E. Bogaers, S. Kok, B. D. Reddy, and T. Franz. “Quasi-Newton methods for implicit black-box FSI coupling”. In: *Computer Methods in Applied Mechanics and Engineering* 279 (2014), pp. 113–132.

[18] E. H. van Brummelen. “Added mass effects of compressible and incompressible flows in fluid-structure interaction”. In: (2009).

[19] M Brun, A Gravouil, A Combescure, and A Limam. “Two FETI-based heterogeneous time step coupling methods for Newmark and α-schemes derived from the energy method”. In: *Computer methods in applied mechanics and engineering* 283 (2015), pp. 130–176.

[20] H.-J. Bungartz, F. Lindner, B. Gatzhammer, M. Mehl, K. Scheufele, A. Shukaev, and B. Uekermann. “preCICE–a fully parallel library for multi-physics surface coupling”. In: *Computers & Fluids* 141 (2016).

[21] F. Casadei and N. Leconte. “Coupling finite elements and finite volumes by Lagrange multipliers for explicit dynamic fluid–structure interaction”. In: *International Journal for Numerical Methods in Engineering* 86.1 (Apr. 2011).

[22] L. Cheung Yau. “Conjugate Heat Transfer with the Multiphysics Coupling Library preCICE”. In: *Technical University of Munich* (2016).

[23] S. K. Chimakurthi, S. Reuss, M. Tooley, and S. Scampoli. “ANSYS Workbench System Coupling: a state-of-the-art computational framework for analyzing multiphysics problems”. In: *engineering with Computers* 34.2 (2018), pp. 385–411.

[24] G. Chourdakis. “A general OpenFOAM adapter for the coupling library preCICE”. Masterarbeit. Technical University of Munich, 2017.

[25] J. Chung and G. M. Hulbert. “A Time Integration Algorithm for Structural Dynamics With Improved Numerical Dissipation: The Generalized- Method”. In: *Journal of Applied Mechanics* 60.2 (1993), pp. 371–375. ISSN: 0021-8936. (Visited on 05/11/2022).

[26] É. Clapeyron. “Mémoire sur la puissance motrice de la chaleur”. In: *Journal de l’École polytechnique* 14 (1834), pp. 153–190.

[27] A. Combescure and A. Gravouil. “A numerical scheme to couple subdomains with different time-steps for predominantly linear transient analysis”. In: *Computer Methods in Applied Mechanics and Engineering* 191.11 (2002), pp. 1129–1157.

[28] A. Combescure, A. Gravouil, and B Herry. “An algorithm to solve transient structural non-linear problems for non-matching time-space domains”. In: *Computers & structures* 81.12 (2003), pp. 1211–1222.

[29] A. De Boer, A. H. van Zuijlen, and H. Bijl. “Review of coupling methods for non-matching meshes”. In: *Computer Methods in Applied Mechanics and Engineering*. Domain Decomposition Methods: recent advances and new challenges in engineering 196.8 (Jan. 2007).

[30] J. Degroote, K.-J. Bathe, and J. Vierendeels. “Performance of a new partitioned procedure versus a monolithic procedure in fluid–structure interaction”. In: *Computers & Structures* 87.11-12 (2009), pp. 793–801.

[31] F. Del Pin and I. Çaldichoury. “LS-DYNA® 980: Recent Developments, Application Areas and Validation Process of the Incompressible fluid solver (ICFD) in LS-DYNA Part”. In: *13th International LS-DYNA Users Conference* (2016).

[32] A. Demir, A. E. Dincer, Z. Bozkus, and A. S. Tijsseling. “Numerical and experimental investigation of damping in a dam-break problem with fluid-structure interaction”. In: *Journal of Zhejiang University-SCIENCE A* 20.4 (2019), pp. 258–271.

[33] G. Dhondt. *The finite element method for three-dimensional thermomechanical applications*. John Wiley & Sons, 2004.

[34] D. Dureisseix. “Une approche multi-échelles pour des calculs de structures sur ordinateurs à architecture parallèle”. PhD thesis. École normale supérieure de Cachan-ENS Cachan, 1997.

[35] G Duvaut and J. Lions. “Les inéquations variationnelles en Mécanique et en Physique”. In: *Dunod, Paris* (1972).

[36] A. F. Emery. “An evaluation of several differencing methods for inviscid fluid flow problems”. In: *Journal of Computational Physics* 2.3 (1968).

[37] L. Euler. “Principes généraux du mouvement des fluides”. In: *Mémoires de l’Académie des Sciences de Berlin* (1757), pp. 274–315.

[38] C. Farhat. “A saddle-point principle domain decomposition method for the solution of solid mechanics problems”. In: *Domain Decomposition Methods for Partial Differential Equations* (1992), pp. 271–292.

[39] C. Farhat, L. Crivelli, and F.-X. Roux. “A transient FETI methodology for large-scale parallel implicit computations in structural mechanics”. In: *International Journal for Numerical Methods in Engineering* 37.11 (1994), pp. 1945–1975.

[40] C. Farhat, J. Mandel, and F. X. Roux. “Optimal convergence properties of the FETI domain decomposition method”. In: *Computer methods in applied mechanics and engineering* 115.3-4 (1994), pp. 365–385.

[41] C. Farhat and F.-X. Roux. “A method of finite element tearing and interconnecting and its parallel solution algorithm”. In: *International Journal for Numerical Methods in Engineering* 32.6 (1991).

[42] C. Farhat, K. G. van der Zee, and P. Geuzaine. “Provably second-order time-accurate loosely-coupled solution algorithms for transient nonlinear computational aeroelasticity”. In: *Computer Methods in Applied Mechanics and Engineering*. Fluid-Structure Interaction 195.17 (Mar. 2006).

[43] F. F. Felker. “Direct solution of two-dimensional Navier-Stokes equations for static aeroelasticity problems”. In: *AIAA journal* 31.1 (1993).

[44] A. Fernier, V. Faucher, and O. Jamond. “Multi-model Arlequin approaches for fast transient, FSI-oriented, fluid dynamics with explicit time integration”. In: *Computers & Fluids* 199 (2020), p. 104428.

[45] A. Franci, E. Oñate, and J. M. Carbonell. “Unified Lagrangian formulation for solid and fluid mechanics and FSI problems”. In: *Computer Methods in Applied Mechanics and Engineering* 298 (2016), pp. 520–547.

[46] R. M. Franck and R. B. Lazarus. “Mixed eulerian-lagrangian method”. In: *Methods in computational physics* 3 (1964), pp. 47–67.

[47] B. Gatzhammer. “Efficient and flexible partitioned simulation of fluid-structure interactions”. PhD thesis. Technische Universität München, 2014.

[48] M. Géradin and D. Rixen. *Théorie des vibrations: application à la dynamique des structures*. Vol. 2. Masson Paris, 1993.

[49] P Germain. “Cours de Mécanique des milieux continus. Tomes 1 : théorie générale”. In: *Ellipses, Paris* (1973).

[50] A. Ghanem, M. Torkhani, N. Mahjoubi, T. Baranger, and A. Combescure. "Arlequin framework for multi-model, multi-time scale and heterogeneous time integrators for structural transient dynamics". In: *Computer methods in applied mechanics and engineering* 254 (2013), pp. 292–308.

[51] S. Godunov and I Bohachevsky. "Finite difference method for numerical computation of discontinuous solutions of the equations of fluid dynamics". In: *Matematičeskij sbornik* 47.3 (1959), pp. 271–306.

[52] A. Gravouil, A. Combescure, and M. Brun. "Heterogeneous asynchronous time integrators for computational structural dynamics". In: *International Journal for Numerical Methods in Engineering* 102.3-4 (Apr. 2015).

[53] A. Gravouil and A. Combescure. "Multi-time-step explicit–implicit method for non-linear structural dynamics". In: *International Journal for Numerical Methods in Engineering* 50.1 (Jan. 2001).

[54] C. J. Greenshields, H. G. Weller, L. Gasparini, and J. M. Reese. "Implementation of semi-discrete, non-staggered central schemes in a colocated, polyhedral, finite volume framework, for high-speed viscous flows". In: *International Journal for Numerical Methods in Fluids* (2009). (Visited on 05/07/2020).

[55] B Herry, L Di Valentin, and A. Combescure. "An approach to the connection between subdomains with non-matching meshes for transient mechanical analysis". In: *International Journal for Numerical Methods in Engineering* 55.8 (2002), pp. 973–1003.

[56] G. Hou, J. Wang, and A. Layton. "Numerical Methods for Fluid-Structure Interaction — A Review". In: *Communications in Computational Physics* 12.2 (Aug. 2012).

[57] T. J. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Courier Corporation, 2012.

[58] A. Ibrahimbegovic, C. Kassiotis, and R. Niekamp. "Fluid-Structure Interaction Problems Solution by Operator Split Methods and Efficient Software Development by Code-Coupling". In: *Coupled System Mechanics* 5.2 (2016).

[59] F Ischinger, M. Anthonissen, and B Koren. "A monolithic fluid-structure interaction method, application to a piston problem". In: *ECCOMAS Congress 2016* (2016).

[60] M. Joosten, W. Dettmer, and D Perić. "Analysis of the block Gauss–Seidel solution procedure for a strongly coupled model problem with reference to fluid–structure interaction". In: *International Journal for Numerical Methods in Engineering* 78.7 (2009), pp. 757–778.

[61] A. Karimi, R. Razaghi, H. Biglari, T. Sera, and S. Kudo. “Collision of the glass shards with the eye: A computational fluid-structure interaction model”. In: *Journal of chemical neuroanatomy* 90 (2018), pp. 80–86.

[62] A. Karimi, R. Razaghi, M. Navidbakhsh, T. Sera, and S. Kudo. “Computing the influences of different Intraocular Pressures on the human eye components using computational fluid-structure interaction model”. In: *Technology and Health Care* 25.2 (2017), pp. 285–297.

[63] A. Khe, A. Cherevko, A. Chupakhin, M. Bobkova, A. Krivoshapkin, and K. Y. Orlov. “Haemodynamics of giant cerebral aneurysm: A comparison between the rigid-wall, one-way and two-way FSI models”. In: *Journal of Physics: Conference Series* 722.1 (2016), p. 012042.

[64] W. Kim and H. Choi. “Immersed boundary methods for fluid-structure interaction: A review”. In: *International Journal of Heat and Fluid Flow* 75 (2019), pp. 301–309.

[65] A Korobenko, M.-C. Hsu, I Akkerman, J Tippmann, and Y. Bazilevs. “Structural mechanics modeling and FSI simulation of wind turbines”. In: *Mathematical Models and Methods in Applied Sciences* 23.02 (2013), pp. 249–272.

[66] D Kowollik, V Tini, S Reese, and M Haupt. “3D fluid–structure interaction analysis of a typical liquid rocket engine cycle based on a novel viscoplastic damage model”. In: *International journal for numerical methods in engineering* 94.13 (2013), pp. 1165–1190.

[67] A. Kurganov and E. Tadmor. “New High-Resolution Central Schemes for Nonlinear Conservation Laws and Convection–Diffusion Equations”. In: *Journal of Computational Physics* 160.1 (May 2000), pp. 241–282.

[68] W. Kutta. “Beitrag zur naherungsweisen integration totaler differentialgleichungen”. In: *Z. Math. Phys.* 46 (1901), pp. 435–453.

[69] A. Larsen. “Aerodynamics of the Tacoma Narrows Bridge-60 years later”. In: *Structural Engineering International* 10.4 (2000), pp. 243–248.

[70] P. Le Tallec and M. Vidrascu. “Generalized Neumann-Neumann preconditioners for iterative substructuring”. In: *Domain Decomposition Methods in Sciences and Engineering*. 1996.

[71] E. Lefrançois and J.-P. Boufflet. “An introduction to fluid-structure interaction: application to the piston problem”. In: *SIAM review* 52.4 (2010), pp. 747–767.

[72] E. Lefrançois, G Dhatt, and D Vandromme. “Fluid–structural interaction with application to rocket engines”. In: *International journal for numerical methods in fluids* 30.7 (1999).

[73] Z. Li, J. Leduc, J. Nunez-Ramirez, A. Combescure, and J.-C. Marongiu. "A non-intrusive partitioned approach to couple smoothed particle hydrodynamics and finite element methods for transient fluid-structure interaction problems with large interface motion". In: *Computational Mechanics* 55.4 (Apr. 2015).

[74] F. Lindner. "Data transfer in partitioned multi-physics simulations: interpolation & communication". In: (2019).

[75] G. Link, M Kaltenbacher, M. Breuer, and M Döllinger. "A 2d finite-element scheme for fluid–solid–acoustic interactions and its application to human phonation". In: *Computer Methods in Applied Mechanics and Engineering* 198.41-44 (2009).

[76] N. Mahjoubi, A. Gravouil, and A. Combescure. "Coupling subdomains with heterogeneous time integrators and incompatible time steps". In: *Computational mechanics* 44.6 (2009), pp. 825–843.

[77] M Malvè, A. P. Del Palomar, J. López-Villalobos, A Ginel, and M Doblaré. "FSI analysis of the coughing mechanism in a human trachea". In: *Annals of biomedical engineering* 38.4 (2010), pp. 1556–1565.

[78] J. Mandel. "Balancing domain decomposition". In: *Communications in numerical methods in engineering* 9.3 (1993), pp. 233–241.

[79] J. Mandel and M. Brezina. "Balancing domain decomposition for problems with large jumps in coefficients". In: *Mathematics of Computation* 65.216 (1996), pp. 1387–1401.

[80] M. Mayr, T. Kloppel, W. A. Wall, and M. W. Gee. "A temporal consistent monolithic approach to fluid-structure interaction enabling single field predictors". In: *SIAM Journal on Scientific Computing* 37.1 (2015).

[81] S Meduri, M Cremonesi, U Perego, O Bettinotti, A Kurkchubasche, and V Oancea. "A partitioned fully explicit Lagrangian finite element method for highly nonlinear fluid-structure interaction problems". In: *International Journal for Numerical Methods in Engineering* 113.1 (2018).

[82] C Michler, S. Hulshoff, E. Van Brummelen, and R. De Borst. "A monolithic approach to fluid–structure interaction". In: *Computers & fluids* 33.5-6 (2004), pp. 839–848.

[83] R. Mittal and G. Iaccarino. "Immersed boundary methods". In: *Annu. Rev. Fluid Mech.* 37 (2005), pp. 239–261.

[84] A. O. Mohmmed, H. H. Al-Kayiem, A. Osman, and O. Sabir. "One-way coupled fluid–structure interaction of gas–liquid slug flow in a horizontal pipe: Experiments and simulations". In: *Journal of Fluids and Structures* 97 (2020), p. 103083.

[85] S. A. Morton, R. B. Melville, and M. R. Visbal. “Accuracy and coupling issues of aeroelastic Navier-Stokes solutions on deforming meshes”. In: *Journal of Aircraft* 35.5 (1998).

[86] N. M. Newmark. “A method of computation for structural dynamics”. In: *Journal of the engineering mechanics division* 85.3 (1959), pp. 67–94.

[87] F. Nobile. “Coupling strategies for the numerical simulation of blood flow in deformable arteries by 3D and 1D models”. In: *Mathematical and Computer Modelling*. Trends in Application of Mathematics to Medicine 49.11 (June 2009).

[88] W. F. Noh. *CEL: A time-dependent, two-space-dimensional, coupled Eulerian-Lagrange code*. Tech. rep. Lawrence Radiation Lab., Univ. of California, Livermore, 1963.

[89] J. Nunez-Ramirez, J.-C. Marongiu, M. Brun, and A. Combescure. “A partitioned approach for the coupling of SPH and FE methods for transient nonlinear FSI problems with incompatible time-steps”. In: *International Journal for Numerical Methods in Engineering* 109.10 (2017).

[90] I. Papagiannis, Y. Elias, M. Tigges, and T. Mueller. “Comparison of one-and two-way coupling of Fluid-Structure Interaction model of peristaltic pump”. In: ().

[91] K. Park, C. Felippa, and J. DeRuntz. “Stabilization of staggered solution procedures for fluid-structure interaction analysis”. In: *Computational methods for fluid-structure interaction problems* 26.94-124 (1977).

[92] C. S. Peskin. “Flow patterns around heart valves: a numerical method”. In: *Journal of computational physics* 10.2 (1972), pp. 252–271.

[93] S. Piperno, C. Farhat, and B. Larrouturou. “Partitioned procedures for the transient solution of coupled aroelastic problems Part I: Model problem, theory and two-dimensional application”. In: *Computer methods in applied mechanics and engineering* 124.1-2 (1995), pp. 79–112.

[94] D. Pitilakis, M. Dietz, D. M. Wood, D. Clouteau, and A. Modaressi. “Numerical simulation of dynamic soil–structure interaction in shaking table testing”. In: *Soil dynamics and earthquake Engineering* 28.6 (2008), pp. 453–467.

[95] A Prakash and K. Hjelmstad. “A FETI-based multi-time-step coupling method for Newmark schemes in structural dynamics”. In: *International journal for numerical methods in engineering* 61.13 (2004), pp. 2183–2204.

[96] A. Prakash. *Multi-time-step domain decomposition and coupling methods for non-linear structural dynamics*. Vol. 68. 11. 2007.

[97] S. Price. “A review of theoretical models for fluidelastic instability of cylinder arrays in cross-flow”. In: *Journal of Fluids and Structures* 9.5 (1995), pp. 463–518.

[98] L.-D. R13. *LS-DYNA KEYWORD USER’S MANUAL*. 2021.

[99] S. M. Rifai, Z. Johan, W.-P. Wang, J.-P. Grisval, T. J. Hughes, and R. M. Ferencz. “Multiphysics simulation of flow-induced vibrations and aeroelasticity on parallel computing platforms”. In: *Computer methods in applied mechanics and engineering* 174.3-4 (1999).

[100] P. L. Roe. “Approximate Riemann solvers, parameter vectors, and difference schemes”. In: *Journal of computational physics* 43.2 (1981), pp. 357–372.

[101] M. R. Ross, M. A. Sprague, C. A. Felippa, and K. C. Park. “Treatment of acoustic fluid–structure interaction by localized Lagrange multipliers and comparison to alternative interface-coupling methods”. In: *Computer Methods in Applied Mechanics and Engineering* 198.9 (2009).

[102] C. Runge. “Über die numerische Auflösung von Differentialgleichungen”. In: *Mathematische Annalen* 46.2 (1895), pp. 167–178.

[103] P. B. Ryzhakov, R. Rossi, S. R. Idelsohn, and E. Oñate. “A monolithic Lagrangian approach for fluid–structure interaction problems”. In: *Computational Mechanics* 46.6 (2010).

[104] G. Santo, M. Peeters, W. Van Paepegem, and J. Degroote. “Comparison between a Chimera technique and sliding interfaces for fluid-structure interaction simulations of wind turbines”. In: *Wind Energy Science Conference 2017 (WESC2017)*. 2017, pp. 266–266.

[105] M Sayed, T. Lutz, E Krämer, S. Shayegan, and R Wüchner. “Aeroelastic analysis of 10 MW wind turbine using CFD–CSD explicit FSI-coupling approach”. In: *Journal of Fluids and Structures* 87 (2019), pp. 354–377.

[106] K. Scheufele. “Coupling schemes and inexact Newton for multi-physics and coupled optimization problems”. In: (2018).

[107] Y. Shimizu, A. Khayyer, and H. Gotoh. “An SPH-based fully-Lagrangian mesh-free implicit FSI solver with high-order discretization terms”. In: *Engineering Analysis with Boundary Elements* 137 (2022), pp. 160–181.

[108] G. Szabó, J. Györgyi, and G. Kristóf. “Three-dimensional FSI Simulation by Using a Novel Hybrid Scaling–Application to the Tacoma Narrows Bridge”. In: *Periodica Polytechnica Civil Engineering* 64.4 (2020), pp. 975–988.

[109] K. Takizawa, T. E. Tezduyar, C. Boswell, Y. Tsutsui, and K. Montel. “Special methods for aerodynamic-moment calculations from parachute FSI modeling”. In: *Computational Mechanics* 55.6 (2015), pp. 1059–1069.

[110] P. L. Tallec. “Domain Decomposition Methods in Computational Mechnaics”. In: *Computational Mechnaics Advances* 1 (1994), pp. 121–220.

[111] T. E. Tezduyar, K. Takizawa, T. Brummer, and P. R. Chen. “Space–time fluid–structure interaction modeling of patient-specific cerebral aneurysms”. In: *International Journal for Numerical Methods in Biomedical Engineering* 27.11 (2011), pp. 1665–1710.

[112] C. Trivedi. “A review on fluid structure interaction in hydraulic turbines: A focus on hydrodynamic damping”. In: *Engineering Failure Analysis* 77 (2017), pp. 1–22.

[113] B. Uekermann, H.-J. Bungartz, L. C. Yau, G. Chourdakis, and A. Rusch. “Official preCICE adapters for standard open-source solvers”. In: *Proceedings of the 7th GACM Colloquium on Computational Mechanics for Young Scientists from Academia*. 2017.

[114] B. W. Uekermann. “Partitioned fluid-structure interaction on massively parallel systems”. PhD thesis. Technische Universität München, 2016.

[115] V. Vuorinen, J.-P. Keskinen, C. Duwig, and B. J. Boersma. “On the implementation of low-dissipative Runge–Kutta projection methods for time dependent flows using OpenFOAM®”. In: *Computers & Fluids* 93 (2014), pp. 153–163.

[116] W. A. Wall and T. Rabczuk. “Fluid–structure interaction in lower airways of CT-based lung geometries”. In: *International Journal for Numerical Methods in Fluids* 57.5 (2008), pp. 653–675.

[117] H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. “A tensorial approach to computational continuum mechanics using object-oriented techniques”. In: *Computers in physics* 12.6 (1998).

[118] J. Wood, G. De Nayer, and M. Schmidt S.and Breuer. “Experimental investigation and large-eddy simulation of the turbulent flow past a smooth and rigid hemisphere”. In: *Flow, Turbulence and Combustion* 97.1 (2016), pp. 79–119.

[119] J. Wu, Y. Cheng, W. Zhou, C. Zhang, and W. Diao. “GPU acceleration of FSI simulations by the immersed boundary-lattice Boltzmann coupling scheme”. In: *Computers & mathematics with applications* 78.4 (2019), pp. 1194–1205.

[120] G Yagawa and R Shioya. “Parallel finite elements on a massively parallel computer with domain decomposition”. In: *Computing Systems in Engineering* 4.4-6 (1993), pp. 495–503.

[121] Z.-c. Zhang, G. Cook Jr, and K.-s. Im. “Overview of the CESE Compressible Fluid and FSI Solvers”. In: *16th International LS-DYNA Users Conference* (2020).

FOLIO ADMINISTRATIF

Thèse de l'Université de Lyon opérée au sein de l'INSA de Lyon

NOM : Gibert **DATE DE SOUTENANCE :** 22/07/2022

PRENOM : Marie

TITRE : A new robust co-simulation approach for transient Fluid-Structure Interaction problems

NATURE : Doctorat **NUMERO D'ORDRE :** 2022LYSEI071

ECOLE DOCTORALE : MEGA

SPECIALITE : Génie mécanique

RESUME :

Dans ce travail, nous proposons une méthode de couplage pour résoudre des problèmes transitoire d'interactions fluide-structure (FSI), basée sur une formulation monolithique et résolue par un algorithme de co-simulation. L'objectif principal étant de proposer une méthode conservative, mais aussi d'utiliser des solveurs existants pour la simulation des sous-domaines fluide et structure.

La formulation monolithique du problème de couplage utilise une approche de Schur duale pour imposer la continuité des vitesses normales à l'interface. Puis, l'algorithme de simulation est basé sur l'extension de la méthode GC au problème FSI, afin de coupler des méthodes de discrétisation temporelles et spatiales hétérogènes ainsi que des échelles de temps différentes. Le sous-domaine structure est discrétisé par la méthode des éléments finis et un schéma d'intégration de Newmark implicite. Le sous-domaine fluide quant à lui, est discrétisé par la méthode des volumes finis centrés sur les cellules et un schéma d'intégration explicite de Runge-Kutta. Enfin, chaque sous-domaine est piloté par une échelle de temps qui lui est propre.

La méthode proposée est validée sur un cas test académique. Puis elle est intégrée dans la librairie de couplage preCICE, afin de simuler des problèmes FSI en utilisant un solveur de dynamique des structures et un solveur de CFD existants. Ainsi, deux problèmes de référence, *forward step* et *perpendicular flap*, sont résolus en utilisant les logiciels OpenFOAM et CalculiX.

MOTS-CLEFS : Interaction Fluide-Strucure, méthodes énérgétique, multi-échelle temporelle, éléments finis, volumes finis, couplage de code

LABORATOIRE DE RECHERCHE : Lamcos

DIRECTEUR DE THESE : Anthony Gravouil

PRESIDENT.E DE JURY : XXX

COMPOSITION DU JURY : Thouraya Baranger, Régis Cottereau, Bing Tie, Sébastien Roth, Anthony Gravouil, Michael Brun