Goal-oriented metric-based mesh adaptation for unsteady CFD simulations involving moving geometries

Éléonore Gauci

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES FONDAMENTALES ET APPLIQUÉES

THÈSE DE DOCTORAT

Présentée par

Éléonore Gauci

Pour obtenir le grade de Docteur de l'Université Côte d'Azur

Spécialité: Sciences pour l'ingénieur

Adaptation de maillage orientée fonctionnelle et basée sur une métrique pour des simulations aérodynamiques en géométrie variable/Goal-oriented metric-based mesh adaptation for unsteady CFD simulations involving moving geometries

---

soutenue le Mercredi 12 Décembre 2018

devant le jury composé de:

Rapporteurs :	M. Bruno Koobus	-	Université de Montpellier
	Mme Maria Vittoria Salvetti	-	Université de Pise
Examinateurs :	Mme Anca Belme	-	Sorbonne Université
	M. Boniface Nkonga	-	Université Côte d'Azur
Invité :	M. Olivier Allain	-	Société Lemma
Directeurs :	M. Frédéric Alauzet	-	Inria Saclay
	M. Alain Dervieux	-	Inria Sophia-Antipolis

Goal-oriented metric-based mesh adaptation for unsteady CFD simulations involving moving geometries

Abstract When dealing with CFD problems, mesh adaptation is interesting for its ability to approach the asymptotic convergence and to obtain an accurate prediction for complex flows at a lower cost. Anisotropic mesh adaptation method reduces the number of degrees of freedom required to reach a given solution accuracy, thus impact favorably the CPU time. Moreover, it reduces the numerical scheme dissipation by automatically taking into account the anisotropy of the physical phenomena inside the mesh. Two main approaches exist in the literature. Feature-based mesh adaptation which is mainly deduced from an interpolation error estimate using the Hessian of the chosen sensor controls the interpolation error of the sensor over the whole computational domain. Such approach is easy to set-up and has a wide range of application, but it does not take into account the considered PDE used to solve the problem. On the other hand, goal-oriented mesh adaptation, which focuses on a scalar output function, takes into consideration both the solution and the PDE in the error estimation thanks to the adjoint state. But, the design of such error estimate is much more complicated. This thesis presents the results obtained with different CFD methods : the Arbitrary Lagrangian Eulerian (ALE) flow solvers with explicit and implicit schemes are presented and coupled to the moving mesh process, the feature-based unsteady mesh adaptation for moving geometries takes into account the changes of connectivites during the whole simulation, the adjoint state is extended to moving geometries problems and goal-oriented unsteady mesh adaptation for moving meshes is derived from an a priori error estimate. Several numerical examples are considered in the aeronautics sector and the field of civil security.

Keywords Metric-based mesh adaptation, anisotropy, adjoint, unsteady simulations, moving mesh, ALE

Adaptation de maillage orientée fonctionnelle et basée sur une métrique pour des simulations aérodynamiques en géométrie variable

---

Résumé En ce qui concerne les problèmes de Dynamique des Fluides Numériques, l'adaptation du maillage est intéressante pour sa capacité à aborder la convergence asymptotique et à obtenir une prévision précise pour des flux complexes à moindre coût. La méthode d'adaptation de maillage anisotrope réduit le nombre de degrés de liberté nécessaires pour atteindre la précision d'une solution donnée, ce qui a un impact positif sur le temps de calcul. De plus, il réduit la dissipation du schéma numérique en tenant compte automatiquement de l'anisotropie des phénomènes physiques à l'intérieur du maillage. Deux approches principales existent dans la littérature. L'adaptation du maillage basée sur les caractéristiques géométriques, qui est principalement déduite d'une estimation de l'erreur d'interpolation utilisant la hessienne du senseur choisi, contrôle l'erreur d'interpolation du capteur sur l'ensemble du domaine de calcul. Une telle approche est facile à mettre en place et a un large éventail d'applications, mais elle ne prend pas en compte l'EDP considérée utilisée pour résoudre le problème. D'autre part, l'adaptation de maillage orientée fonctionnelle, qui se concentre sur une fonctionnelle scalaire, prend en compte à la fois la solution et l'EDP dans l'estimation d'erreur grâce à l'état adjoint. Mais, la conception de cette estimation d'erreur est beaucoup plus compliquée. Cette thèse présente les résultats obtenus avec différentes méthodes de Dynamique des Fluides Numériques: les solveurs de flux arbitrairement lagrangiens-eulériens (ALE) avec schémas explicites et implicites sont présentés et couplés au mouvement de maillage, l'adaptation de maillage feature-based instationnaire pour les géométries mobiles prend en compte les changements des connectivités de maillage durant toute la simulation, l'état adjoint est étendu aux problèmes de géométries mobiles et l'adaptation de maillage instationnaire orientée fonctionnelle pour les maillages mobiles est déduite d'une estimation d'erreur a priori. Plusieurs exemples numériques issus du secteur aéronautique et du domaine de sécurité civile sont considérés.

Mots-Clés Adaptation de maillage basée métrique, anisotropie, adjoint, simulations instationnaires, mouvement de maillage, ALE

"If you're offered a seat on a rocket ship,
don't ask what seat ! Just get on."

Contents

Convention **3**

Introduction **5**

1 State of the art of previous works **11**

1.1 Continuous mesh framework 11

1.1.1 Duality between discrete and continuous element : notion of unit element 11

1.1.2 Duality between discrete and continuous mesh : notion of unit mesh . . . 17

1.1.3 Duality between discrete and continuous interpolation error: notion of continuous linear interpolate 18

1.1.4 Summary 22

1.2 Mesh adaptation 22

1.2.1 Steady mesh adaptation 22

1.2.2 Unsteady mesh adaptation 30

1.3 Mesh adaptation for moving geometries 36

1.3.1 Mesh adaptation problem 36

1.3.2 Optimal instantaneous ALE continuous mesh minimizing the interpolation error in L^p norm: the ALE metric 37

1.3.3 Space-time error analysis for dynamic meshes 39

1.3.4 Mesh adaptation algorithm 40

1.4 Conclusion 41

2 Resolution of Euler equations in the ALE framework **43**

2.1 Mesh-connectivity-change moving mesh strategy 45

2.1.1 Linear elasticity mesh deformation method 46

2.1.2 Improving mesh deformation algorithm efficiency 47

2.1.3 Local mesh optimization 48

2.1.4 Handling of boundaries 50

2.1.5 Moving mesh algorithm 50

2.1.6 Moving mesh time steps 52

2.2 Arbitrary Lagrangian Eulerian flow solver 52

2.2.1 Euler equations in the ALE framework 53

2.2.2 Spatial discretization 53

Rusanov Riemann solver/Roe Riemann solver/HLLC Riemann solver 57

2.2.3 Time discretization . . . 64
2.3 FSI coupling . . . 78
2.4 Numerical experiments . . . 79
2.4.1 Naca0012 Pitching . . . 79
2.4.2 FSI Case : 2D Bomb drop . . . 83
2.4.3 3D Nosing-up F117 . . . 87
2.5 Conclusion . . . 90

3 Hessian-based mesh adaptation for moving geometries 93
3.1 Error estimate for unsteady simulations . . . 95
3.1.1 Error model . . . 96
3.1.2 Spatial minimization for a fixed t . . . 96
3.1.3 Temporal minimization . . . 97
3.1.4 Error analysis for time sub-intervals . . . 98
3.1.5 Global fixed-point mesh adaptation algorithm for unsteady simulation . . . 100
3.2 From theory to practice . . . 102
3.2.1 Computation of the Hessian metric . . . 102
3.2.2 Choice of the mean Hessian-metric . . . 104
3.2.3 Choice of the optimal continuous mesh . . . 105
3.2.4 Matrix-free P1-exact conservative solution transfer . . . 106
3.2.5 The remeshing step . . . 107
3.2.6 Software used . . . 108
3.3 Unsteady mesh adaptation for dynamic meshes . . . 108
3.4 Handling the swaps : update of the adaptation algorithm . . . 110
3.4.1 Metric for optimizations . . . 110
3.4.2 Modification of the algorithm . . . 111
3.5 Validation with analytic solutions of the metric for optimizations . . . 113
3.5.1 Procedure . . . 113
3.5.2 Analytic function considered . . . 114
3.5.3 Numerical 3D example : 3D F117 nosing up . . . 116
3.6 Conclusion . . . 119

4 Adjoint Methods 121
4.1 State of the art of adjoint method . . . 122
4.1.1 Mathematical point of view . . . 122
4.1.2 Interpretation . . . 123
4.1.3 Numerical adjoint approaches . . . 123

4.2 Computation of adjoint Euler system in ALE formulation . . . 125
4.2.1 Unsteady adjoint Euler equations in ALE formulation . . . 125
4.2.2 Discrete adjoint state in ALE formulation . . . 125
4.2.3 Implicit scheme . . . 127
4.3 Connectivity-change moving mesh strategy . . . 128
4.4 The Discrete Geometric Conservation Law . . . 129
4.5 Backward Implementation . . . 130
4.5.1 Numerical flux computation . . . 130
4.5.2 Memory management . . . 132
4.6 Dealing with memory . . . 132
4.6.1 Packing and interpolation . . . 132
4.6.2 Parallelization . . . 132
4.7 Validation of the adjoint code . . . 133
4.8 Numerical examples . . . 134
4.8.1 Case with imposed movement : Naca0012 Rotation . . . 134
4.8.2 FSI Case : 2D Shock-ball interaction in a tube . . . 137
4.8.3 Numerical qualitative verification in 3D : Lohner blast-like on a Bump . . . 139
4.9 Conclusion . . . 141

5 Goal-oriented mesh adaptation for moving mesh simulations 143
5.1 Optimal adjoint-based ALE metric . . . 144
5.1.1 Functional output of interest . . . 144
5.1.2 Continuous adjoint state . . . 144
5.1.3 Formal error analysis . . . 145
5.1.4 Local error analysis . . . 146
5.1.5 Continuous error model . . . 147
5.2 Implementation of goal-oriented mesh adaptation for unsteady problems involving moving geometries . . . 148
5.2.1 Choice of the goal-oriented metric . . . 148
5.2.2 Memory management . . . 149
5.2.3 Algorithm . . . 150
5.3 Numerical test cases . . . 150
5.3.1 Numerical test case : a Lohner blast-like on a 2D Bump . . . 151
5.3.2 Numerical results . . . 152
5.4 Conclusion . . . 153

Bibliography 157

Remerciements

Je tiens à remercier tout d'abord mes directeurs de thèse : Alain Dervieux et Frédéric Alauzet pour m'avoir fait découvrir le monde de la recherche. Ils m'ont proposé un très beau sujet, riche de possibilités. Constamment soucieux de l'avancement de mes travaux, ils ont su me dispenser une formation très complète. Ils m'ont aussi fait confiance lors de mes exposés en conférences et n'ont jamais hésité à me recommander auprès de la communauté scientifique. Leur rigueur, leurs exigences, et leur engagement pour la recherche resteront pour moi des références.

Je suis très honorée que Madame Maria-Vittoria Salvetti et M. Bruno Koobus aient accepté de rapporter mon travail et je les en remercie. Je suis également reconnaissante envers Mme Anca Belme, M. Boniface Nkonga et M. Olivier Allain pour leur participation à mon jury.

Mes premiers remerciements personnels vont vers ma famille. C'est aussi grâce à eux que je me suis lancée dans ce projet de doctorat. Je remercie mes parents de m'avoir transmis le goût du travail et de la témérité qui nous ont permis à mes frères et moi de réussir. Vous nous avez toujours poussés à nous surpasser, au-delà même de vos propres compétences en réussissant également le pari de nous faire vivre une enfance heureuse. Merci à Edouard : ton intelligence, ta droiture et ta gentillesse font que tu as été un vrai modèle de grand frère pour ta petite soeur. Merci à Marc-Olivier pour ton énergie inépuisable, ton envie de toujours bien-faire et nos moments partagés devant un piano. Merci à François-Xavier de m'avoir souvent appris à garder mon sang froid et à me protéger. Merci à Pierre-Alexis qui a su prouver que même les petits frères deviennent des grands. Merci Nanou pour ton énergie, ta force et ton optimisme. Merci à Atti, à Francoise et à Joseph. Merci à Aurélie, Alexandra, Marion. Merci à Chloé et Maëlys : vous voir grandir, vous émerveiller du monde fait de moi une tante et une marraine comblée.

Personne ne peut réussir seul et c'est pour cela que je souhaite remercier chaleureusement tous les membres du projet Gamma. Je suis reconnaissante envers Paul-Louis George, responsable du projet Gamma, de m'avoir accueillie dans l'équipe. Merci à Adrien pour sa rigueur, sa confiance et son éternel enthousiasme, merci à Loïc M. et à ses précieux conseils, merci à Loïc F. avec lequel on a partagé les années Gamma et nos motivations jamais linéaires, merci à Rémi pour ta constante bienveillance, merci à Julien pour nous avoir toujours fait rire, merci à Nicolas d'avoir répondu à mes premières questions sur le maillage mobile, à Victorien pour m'avoir aussi vite intégrée, à Olivier pour nos duos improvisés basse-piano, à Alexis pour sa constante bonne humeur. Thank you to Dave Marcum and his communicative cheerfulness.

J'aimerais également remercié l'équipe Ecuador dans laquelle j'ai passé ma première année

de thèse ainsi que la société Lemma et ses collaborateurs. Je remercie également toutes les personnes que j'ai croisées durant des séminaires ou des conférences pour nos échanges fructueux.

Ma formation mathématique doit beaucoup aux enseignants de l'université de Nice. Je remercie ici les enseignants-chercheurs du laboratoire LJAD mais aussi les équipes pédagogiques avec lesquelles j'ai travaillé à Polytech'Sophia et à la Sorbonne université pour m'avoir accordé leur confiance. Je remercie également les professeurs de mathématiques du lycée Jean de La Fontaine et en particulier Sylvie pour sa disponibilité et son soutien.

Je voudrais remercier mes amies qui ont accepté et compris mon rythme de travail et ont surtout su m'en sortir de temps en temps. En particulier merci à Géraldine pour tes grandes qualités humaines et ton soutien infaillible. Ton regard sur la vie, ta philosophie, ton intelligence et ta spontanéité m'ont beaucoup appris et notre amitié a rendu ces années de thèse plus agréables. Merci Julie pour ton soutien inébranlable et ta disponibilité quasi quotidienne. Nos discussions toujours pleines d'humour et de joie de vivre ont égayé nos années d'agrégation et de doctorat et ont décuplé notre motivation pour atteindre ensemble nos objectifs.

Mener à bien une thèse est une entreprise difficile, être le soutien de l'ombre l'est tout autant. Je dédie mes derniers remerciements à Arnaud qui m'a toujours soutenue cette dernière année.

Convention

Simplicial mesh

Let n be the dimension of the physical space and let $\Omega \subset \mathbb{R}^n$ be the non-discretized physical domain. Ω is an affine space. The canonical basis of its vectorial space is noted $(\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n)$ in general, but notations $(\mathbf{e}_x, \mathbf{e}_y)$ and $(\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$ can also be used in two and three dimensions. Vectors of $\mathbb{R}^n$ are noted in bold font. The coordinate vector of a point of Ω is generally noted $\mathbf{x} = (x_1, \ldots, x_n)$.

The boundary of Ω, noted $\partial\Omega$, is discretized using simplicial elements the vertices of which are located on $\partial\Omega$. In two dimensions, $\partial\Omega$ is discretized with segments while in three dimensions, boundary surfaces $\partial\Omega$ are represented by triangles. The discretized boundary is noted $\partial\Omega_h$ and Ω_h denotes the sub-domain of $\mathbb{R}^n$ having $\partial\Omega_h$ as boundary.

Building a mesh of Ω_h consists in finding a set of *simplicial* elements - triangles in two dimensions, tetrahedra in three dimensions - noted $\mathcal{H}$, satisfying the following properties:

- *Non-degenerescence*: Each simplicial element K of $\mathcal{H}$ is non-degenerated (no flat elements of null area or volume),
- *Covering*: $\Omega_h = \bigcup_{K \in \mathcal{H}} K$,
- *Non-overlapping*: The intersection of the interior of two different elements of $\mathcal{H}$ is empty:

$$\mathring{K}_i \cap \mathring{K}_j = \varnothing, \forall\, K_i,\, K_j \in \mathcal{H},\, i \neq j\,.$$

- *Conformity*: The intersection of two elements is either a vertex, an edge or a face (in three dimensions) or is empty.

The *conformity* hypothesis is adopted here, because the meshes will be used with a Finite-Volume solver which requires the enforcement of this constraint. Besides, it facilitates the handling of data structures on the solver side and also enables to save a consequent amount of CPU time.
Finally, a mesh is said to be *uniform* if all its elements are almost regular (equilateral) and have the same size h.

Notations and orientation conventions

The following notations will be used in this thesis: K is an element of the mesh $\mathcal{H}$, P_i is the vertex of $\mathcal{H}$ having i as global index, $\mathbf{e}_{ij}$ is the edge linking P_i to P_j. The number of elements, vertices and edges are noted N_t, N_v and N_e, respectively.

The vertices and the edges of an element K are also numbered locally. Vertex numbering inside each element is done in a counter-clockwise (or trigonometric) manner, which enables to compute edges/faces outward normals in a systematic way. This numbering, as well as unit outward normals $\mathbf{n}$ and edges/faces orientations are shown for triangles and tetrahedra in Figure 1. Non-normalized normals will be noted $\boldsymbol{\eta}$.

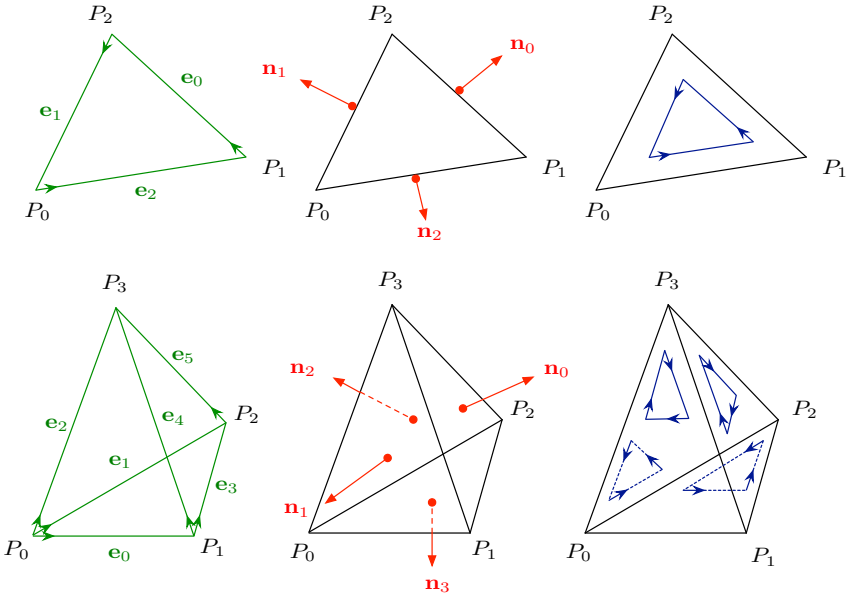


Figure 1: Conventions in a simplicial element K in two (top) and three (bottom) dimensions. Conventions for vertex numbering, edge numbering and orientation (left), unit normal numbering and orientation (middle) and face(s) orientation (right).

Introduction

This manuscript proposes a synthesis of my research during my years of PhD at Inria in Gamma3 and Ecuador teams in the field of anisotropic mesh adaptation applied to unsteady simulations involving moving computational domains.

Often considered as a substitute for modelling systems for which simple closed from analytic solutions are not possible, computer simulations' big advantage is the real ability to generate representive scenarios for which a realisation of all possible states would be prohibitive or impossible in practice. Consequently, numerical simulation has become an integral part of the design process in science and engineering. In sixty years, numerical simulation has come a long way, in conjunction with the development of computing resources. As it became possible to model and simulate more complex problems, industry started to take advantage of its potentiality. However, as the simulation capabilities grow, so do the requirements of both industries and researchers, and a whole set of problems have arisen. In the field of computational fluid dynamics (CFD), the numerical simulations have to deal with ever increasing geometrical and physical complexities. However being able to predict numerically all the features of complex flows around complex geometries remains an unachieved goal. In this sense, scientists and engineers have developed several techniques to make predictions with the maximum of accuracy while mastering the computational efforts in term of CPU time.
In this sense, mesh adaptation methods have been developed to reduce the complexity of the simulations while keeping the same high level of precision. Notably, the increase in computational power allows scientists to not content themselves with steady approximations of the physical phenomena, and to study the effects of time-dependent parameters.

Mesh adaptation for CFD

The computational pipeline for fluid dynamic simulations illustrated by Figure 2 can be summarized as :

$$\textsc{Cad} \rightarrow \textsc{Mesh} \rightarrow \textsc{Solver} \rightarrow \textsc{Visualization/Analysis}$$

The mesh is one of the first step of the computation and as the topological base frame, its role may be crucial. Anisotropic mesh adaptation method reduces the number of degrees of freedom required to reach a given solution accuracy and thus impacts favorably the CPU time. Moreover, it reduces the numerical scheme dissipation by automatically taking into account the anisotropy of the physical phenomena inside the mesh. That's why this technique is nowadays

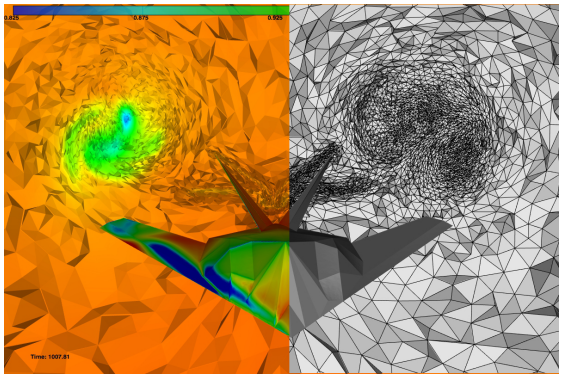


Figure 2: Under the visualization of the numerical solutions of Euler flux solution of f117 in flight *on the Left*, the mesh of the computational volume around which enables to compute the solution *on the Right*

a great deal of interest. Two main approaches exist in the literature. The first one is the feature-based mesh adaptation which is mainly deduced from an interpolation error estimate using the Hessian of the chosen sensor. It controls the error on the sensor over the whole computational domain. Such approach is easy to set-up and has a wide range of application, but it does not take into account the considered PDE used to solve the problem. The second approach is the goal-oriented mesh adaptation. It focus on a scalar output function and takes into consideration both the solution and the PDE in the error estimation thanks to the adjoint state.

In this thesis, we focus on Computational Fluid Dynamics (CFD), and its applications to vorticity in aeronautic and blast mitigations, although the results presented can be used in a vast field of research and industry. The next section presents its numerical context in details.

Numerical context

The present work was conducted in Gamma3 and Ecuador research groups at Inria. Most numerical choices naturally stem from those made in the group, to benefit from both the considerable experience and the valuable software of its members.

- We consider **unstructured meshes** made up of triangles (in 2D) or tetrahedra (in 3D). This geometrical choice is initially due to the fact that it is easier to mesh complex geometries with simplexes.

- We consider **metric-based** mesh adaptation. The theory derives from the classical Riemannian geometry theory and underlies on accurate feature-based or adjoint based error estimates.

- We consider **anisotropic** mesh-adaptation to fit the natural physical phenomena properties inside the mesh in order to improve their representation.

- Regarding moving-boundary meshes, the **body-fitted approach** has been preferred in comparison to other methods for its accurate treatment of boundaries.

- From the aspect of solver, the **Arbitrary-Lagrangian-Eulerian** (ALE) framework has been chosen for its ability to describe physical equations on a mesh moving with an imposed or an abitrary movement.

- Finally, only **two-dimensional or three dimensional mono-fluid problems modelled by the inviscid compressible Euler equations** are considered in this thesis.

Main contributions

During this thesis, I extend **goal-oriented mesh adaptation to time-dependent simulations on moving computational domains**.
My work follows the previous works of [Olivier 2011a, Belme 2011, Barral 2015].

- I contributed to the development and implementation of implicit schemes in ALE formulation for unsteady Euler equation involving moving mesh problems and compared it to the explicit time integration simulations.

- I updated the unsteady feature-based ALE metric to handle mesh optimizations during the computational loop with the error analysis from [N. Barral 2017].

- I extended the adjoint solver of unsteady Euler flows for ALE formulation in two-dimensional (validated) in three-dimensional (partially implemented)

- I added the backward moving mesh to the code.

- I added a new error estimation for goal-oriented mesh adaptation for moving mesh simulations

Organisation and content of thesis

The present thesis is built around two main topics :

- The understanding and improvements of the fluid solver in the ALE framework and the hessian-based mesh adaptation for moving mesh geometries.

- A goal-oriented based anisotropic mesh adaptation for unsteady problems involving moving mesh geometries.

The first three chapters of this work refer to the first point. The remaining parts concentrates on the ALE adjoint solver and the set up of the goal-oriented mesh adaptation.

Chapter 1 The theorical and numerical prerequisites of the next chapters.

Chapter 2 The definition, the implementation and the validation of implicit time integration scheme for the ALE flow solver.

Chapter 3 The completion with respect to several steps for the unsteady mesh adaptation for moving geometries which complement thesis [Barral 2015]. In particular, the importance to update the ALE metric in the flow solver to govern the mesh optimizations during the moving mesh algorithm.

Chapter 4 The development of the unsteady adjoint solver in the ALE framework which requires a backward in time moving mesh algorithm consistent with the forward in time one.

Chapter 5 The time accurate goal-oriented mesh adaptation for moving geometries and its associated error estimate.

Scientific communications

Papers

2015 E. Gauci, F. Alauzet, A. Loseille, A. Dervieux, *Towards goal-oriented mesh adaptation for fluid-structure interaction*, Coupled Problems in Science and Engineering VI 1197 : 1208, Venice, Italy [E. Gauci 2015]

2017 E. Gauci, F. Alauzet and A. Dervieux, *Vortical Flow Prediction of a Moving Aircraft Using Goal-Oriented Anisotropic Mesh Adaptation*, Proc. of 23*rd* AIAA Computational Fluid Dynamics Conference, AIAA-2017-3298, Denver, CO, USA, June 2017Denver, Co, USA [E. Gauci 2017]

Oral Communications

Juin 2017 **AIAA Aviation**, E. Gauci, F. Alauzet and A. Dervieux, *Vortical Flow Prediction of a Moving Aircraft Using Goal-Oriented Anisotropic Mesh Adaptation*, Proc. of 23*rd* AIAA

Computational Fluid Dynamics Conference, AIAA-2017-3298, Denver, CO, USA, June 2017Denver, Co, USA

Nov.2016 **PhD Seminars** *Mesh Adaptation and Fluid Dynamics*, Inria Saclay-LIX and Inria Sophia-Antipolis

Juin 2016 **ECCOMAS Congress 2016**, E. Gauci, F. Alauzet and A. Dervieux, *Goal-oriented mesh adaptation for FSI problems*, ECCOMAS 2016, Hersonissos, Greece, June 2016.

Oct. 2015 **Lebesgue Doctoral Meetings**, *Le maillage pour tous*, Rencontres doctorales Lebesgue, Nantes, France.

Mai 2015 **VI International Conference on Coupled Problems in Science and Engineering**, E. Gauci, F. Alauzet, A. Loseille, A. Dervieux, *Towards goal-oriented mesh adaptation for fluid-structure interaction*, Coupled Problems in Science and Engineering VI 1197 : 1208, Venice, Italy

Mai 2015 **PhD Seminars of Nice** Séminaire des doctorants niçois en mathématiques, St Etienne de Tinée,France

2015-2017 Participations to "Fête de la Science", Inria

Chapter 1

State of the art of previous works

The proposed works in this thesis have been built on three existing theories referred in this chapter. It is quite dense because it corresponds to four thesis : the theoretical foundation have been established in [Loseille 2008] and the three mesh adaptation theories on which this thesis relies have been introduced in [Olivier 2011a] for unsteady feature-based mesh adaptation, in [Belme 2011] for unsteady goal-oriented mesh adaptation and in [Barral 2015] for time-accurate mesh adaptation involving moving geometries.

1.1 Continuous mesh framework

Generating an adapted mesh means optimizing the accuracy of some parts of the simulation domain while de-emphazing other parts we don't need precision. To give rise to this type of mesh we must establish a generator for this assignment. And this assignment must be quasi unique depending on the current localization. At first thought and as it comes to geometry, let's look at the available geometrical tools. Considering a triangle of side 1 in an Euclidian affine space of $\mathbb{R}^2$, its area (i.e. the dot product of one of its pair of vectors) or the angle between these are the same whatever the considered triangle of the space. But generating an unstructured, anisotropic mesh means assigning privileged sizes and local orientations for the elements (triangles or tetrahedra) at each point of the domain. We will see that the use of Riemannian metric spaces is an elegant and efficient way to achieve this goal.
This section is organized as follows. We first succinctly recall the metric-based mesh generation in Sections 1.1.1, 1.1.2, then the Section 1.1.3 gives the theoretical framework of anisotropic mesh adaptation based on the continuous mesh concept.

1.1.1 Duality between discrete and continuous element : notion of unit element

In differential geometry, a metric tensor takes as input a pair of tangent vectors at a point of a surface (or any differentiable manifold) and produces a real scalar number in a way that generalizes many of the familiar properties of the dot product of vectors in Euclidian space.

This continuous point of view enables a practical mathematical representation, and the use of powerful tools of differential geometry.

In the same way as a dot product, metric tensors are used to define the length of and angle between tangent vectors. Let's explain the duality between triangle/tetrahedron and metric tensor, in other terms between discrete and continuous element.

Euclidian space

Let us begin by studying this duality in the Euclidian space.

We consider the vector space $\mathbb{R}^n$, typically $n = 2$ or 3 in our case. A scalar product is a Symmetric Positive Definite (SPD) form. This form can be represented by a SPD matrix $\mathcal{M} = (m_{ij})_{1 \leq i,j \leq n}$, which is called a *metric tensor* or simply *metric*. The scalar product is then written:

$$\begin{array}{rccl} (\cdot\,,\cdot)_{\mathcal{M}}: & \mathbb{R}^n \times \mathbb{R}^n & \longrightarrow & \mathbb{R}^+ \\ & (\mathbf{u}, \mathbf{v}) & \longmapsto & (\mathbf{u},\,\mathbf{v})_{\mathcal{M}} \;=\; \mathbf{u}^T\,\mathcal{M}\,\mathbf{v} \;=\; \displaystyle\sum_{j=1}^{n}\sum_{i=1}^{n} m_{ij} u_i v_j\,. \end{array} \tag{1.1}$$

In the simple case where $\mathcal{M} = \mathcal{I}$ with $\mathcal{I}$ the identity matrix, the scalar product is the canonical Euclidian dot product. A real vector space with the scalar product is called an *Euclidian space*.

The scalar product is useful to compute lengths and angles in the Euclidian space. We can define:

- a distance:

$$\begin{array}{rccl} d_{\mathcal{M}}: & \mathbb{R}^n \times \mathbb{R}^n & \longrightarrow & \mathbb{R}^+ \\ & (P,\,Q) & \longmapsto & d_{\mathcal{M}}\,(P,\,Q) = \sqrt{\overrightarrow{PQ}^T\,\mathcal{M}\,\overrightarrow{PQ}}\,, \end{array} \tag{1.2}$$

  which induces a metric space structure on the vector space,

- and a norm:

$$\begin{array}{rccl} ||\cdot||_{\mathcal{M}}: & \mathbb{R}^n & \longrightarrow & \mathbb{R}^+ \\ & \mathbf{u} & \longmapsto & ||\mathbf{u}||_{\mathcal{M}} = \sqrt{\mathbf{u}^T\,\mathcal{M}\,\mathbf{u}}\,. \end{array} \tag{1.3}$$

From these distance and norm, we deduce the **classic geometrical quantities**:

- the length of an edge $\mathbf{e}$ is given by:

$$\ell_{\mathcal{M}}\,(\mathbf{e}) = \sqrt{\mathbf{e}^T\,\mathcal{M}\,\mathbf{e}}\,, \tag{1.4}$$

- the angle between two vectors $\mathbf{u_1}$ and $\mathbf{u_2}$ is the unique real number $\theta \in [0, \pi]$ such that:

$$\cos\theta = \frac{(\mathbf{u_1},\,\mathbf{u_2})_{\mathcal{M}}}{||\mathbf{u_1}||_{\mathcal{M}}\,||\mathbf{u_2}||_{\mathcal{M}}} \tag{1.5}$$

- the volume of element K is:

$$|K|_{\mathcal{M}} = \sqrt{\det \mathcal{M}}\,|K| \;\text{ where }\; |K| = |K|_{\mathcal{I}}. \tag{1.6}$$

A very useful result on metrics is that $\mathcal{M}$ is **diagonalizable in an orthonormal basis**:

$$\mathcal{M} = \mathcal{R}\,\Lambda\,\mathcal{R}^T,$$

$$\text{where} \begin{cases} \Lambda \;= \text{diag}\,(\lambda_1, \ldots, \lambda_n) \;\text{ is the diagonal matrix made of the eigenvalues of } \mathcal{M}, \\ \mathcal{R} \;= (\,\mathbf{v_1}\,|\,\mathbf{v_2}\,|\,\ldots\,|\mathbf{v_n}\,)^T \;\text{ is the unitary matrix } (i.e.\ \mathcal{R}^T\mathcal{R} = \mathcal{I}) \\ \quad\;\; \text{made of the eigenvectors of } \mathcal{M}. \end{cases} \tag{1.7}$$

We will very often use the **geometric representation** of a metric tensor. In the vicinity $\mathcal{V}(P)$ of point P, the set of points that are at distance ε of P, is given by:

$$\Phi_{\mathcal{M}}(\varepsilon) = \{Q \in \mathcal{V}(P) \mid d_{\mathcal{M}}(Q,P) = \varepsilon\}. \tag{1.8}$$

This relation defines an ellipsoid of center **a** and whose axes are aligned with the eigen directions $\mathbf{v}_i$ of $\mathcal{M}$. The set $\Phi_{\mathcal{M}}(1)$ is called the **unit ball** of $\mathcal{M}$, and the sizes of its axes are $h_i = \lambda_i^{-\frac{1}{2}}$. This unit ball is depicted in Figure 1.1 in two and three dimensions.

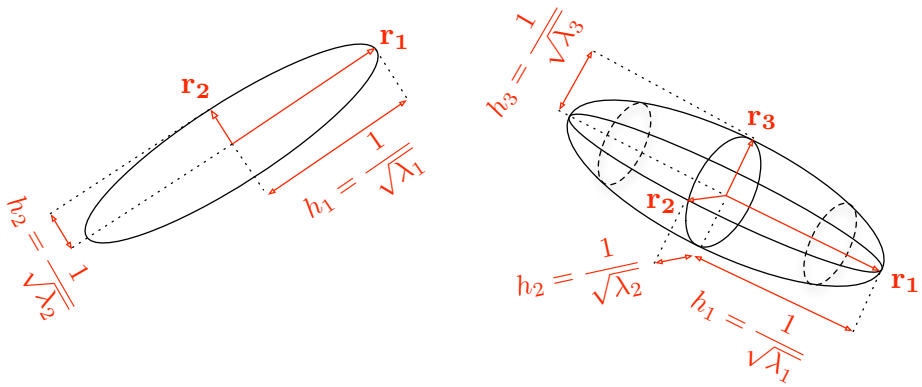


Figure 1.1: *Unit balls associated with metric* $\mathcal{M} = \mathcal{R}\,\Lambda\,\mathcal{R}^T$ *in two and three dimensions.*

A metric tensor $\mathcal{M}$ provides another useful information: the application that maps the unit ball associated with $\mathcal{I}$ to the unit ball associated with $\mathcal{M}$. The application $\Lambda^{\frac{1}{2}}\,\mathcal{R}$ where $\Lambda^{\frac{1}{2}} = \text{diag}(\lambda_i^{\frac{1}{2}})$ defines the mapping from the physical space $(\mathbb{R}^n, \mathcal{I})$ to the Euclidean metric

space $(\mathbb{R}^n, \mathcal{M})$:

$$\begin{aligned} \Lambda^{\frac{1}{2}}\mathcal{R}: \quad (\mathbb{R}^3, \mathcal{I}) &\longmapsto (\mathbb{R}^3, \mathcal{M}) \\ \mathbf{x} &\longrightarrow \left(\Lambda^{\frac{1}{2}}\mathcal{R}\right)\mathbf{x}\,. \end{aligned}$$

This **natural mapping** corresponds to the change of basis from the canonical basis to the orthonormal diagonalization basis of $\mathcal{M}$ and is depicted in Figure 1.2.

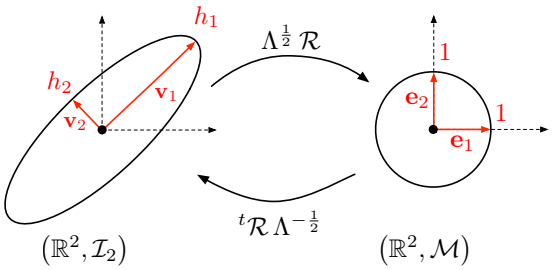


Figure 1.2: *Natural mapping $\Lambda^{\frac{1}{2}}\mathcal{R}$ associated with metric $\mathcal{M}$ in two dimensions. It sends the unit ball of $\mathcal{I}_2$ onto the unit ball of $\mathcal{M}$.*

The key notion of the discrete-continuous duality is the notion of unit element. A tetrahedron K, defined by its list of edges $(\mathbf{e_i})_{i=1..6}$, is said to be an unit element with respect to a metric tensor $\mathcal{M}$ if the length of all its edges is unit in metric $\mathcal{M}$:

$$\forall i = 1, ..., 6, \;\; \ell_{\mathcal{M}}(\mathbf{e_i}) = 1 \text{ with } \ell_{\mathcal{M}}(\mathbf{e_i}) = \sqrt{\mathbf{e_i}^T \mathcal{M}\, \mathbf{e_i}}. \tag{1.9}$$

If K is composed only of unit length edges then its volume $|K|_{\mathcal{M}}$ in $\mathcal{M}$ is constant equal to:

$$|K|_{\mathcal{M}} = \frac{\sqrt{2}}{12} \;\text{ and }\; |K| = \frac{\sqrt{2}}{12}(\det(\mathcal{M}))^{-\frac{1}{2}}, \tag{1.10}$$

where $|K|$ is its Euclidean volume. It is actually used to define classes of equivalence of discrete elements: let $\mathcal{M}$ be a metric tensor, there exists a non-empty infinite set of unit elements with respect to $\mathcal{M}$. Conversely, given an element K such that $|K| \neq 0$, then there is a unique metric tensor $\mathcal{M}$ for which element K is unit with respect to $\mathcal{M}$.

Consequently, a discrete element can be viewed as a discrete representative of an equivalence class formed by all the unit elements with respect to a metric $\mathcal{M}$. Figure 1.3 depicts some unit elements with respect to a metric tensor, which is geometrically represented by its unit-ball. $\mathcal{M}$ denotes the class of equivalence of all the elements which are unit with respect to $\mathcal{M}$ and is called **continuous element**.

All the discrete representatives of a continuous element $\mathcal{M}$ share some common properties, called invariants, which justify the use of this equivalence relation. They connect the geometric

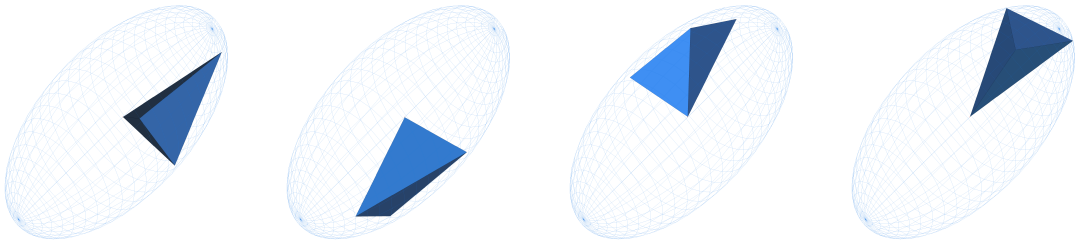


Figure 1.3: *Several unit elements with respect to a metric tensor in 3D.*

properties of the discrete elements to the algebraic properties of metric $\mathcal{M}$. The two main invariants are:

- the edges $\mathbf{e_i}$ of any unit element K with respect to metric $\mathcal{M}$ are unit in $\mathcal{M}$:

$$\forall i = 1, ..., 6, \quad \mathbf{e}_i^T \, \mathcal{M} \, \mathbf{e}_i = 1, \tag{1.11}$$

- conservation of the Euclidean volume for any unit element K with respect to metric $\mathcal{M}$

$$|K| = \frac{\sqrt{3}}{4} \det(\mathcal{M}^{-\frac{1}{2}}) \text{ in 2D} \quad \text{and} \quad |K| = \frac{\sqrt{2}}{12} \det(\mathcal{M}^{-\frac{1}{2}}) \text{ in 3D.} \tag{1.12}$$

  Many other invariants are given in [Loseille 2008].

Riemannian metric space

We've just seen the link between a metric tensor and an unit element. This idea is the starting point to understand how to create a metric-based adapted mesh. This theory, initially introduced in [George 1991], is based on an unit mesh generation in a prescribed Riemannian metric space.

In fact, working in an Euclidian space is no longer sufficient. This is because the scalar product of Euclidian spaces is the same on the whole space, which means that the distance definition is the same for each point of the space (the unit ball is the same everywhere). However, when used to generate adapted meshes with different element sizes in the domain, it is convenient to have a distance that depends on the position of the space. Thus, we now consider a set of SPD tensors $\mathbf{M} = (\mathcal{M}(P))_{P \in \Omega}$, also called metric tensor field, defined on the whole domain $\Omega \subset \mathbb{R}^n$. Locally at point P, $\mathcal{M}(P)$ induces a scalar product on $\mathbb{R}^n \times \mathbb{R}^n$. The vector space, with this new structure, is called a **Riemannian metric space**. In this thesis, we will use the same notation $\mathcal{M}$ to speak of the metric field and of the metric tensor at a given point. Notation $\mathbf{M}$ will only be used if the distinction is necessary for pedagogical purposes.

Remark 1. *Unlike usual Riemannian spaces, there is no notion of manifold in Riemannian metric spaces. However, Riemannian metric spaces can be assimilated to functions representing Cartesian surfaces, and the metric tensor defined for a point of the space is a scalar product on the tangent plane for that point. Figure 1.4 gives an example of a Cartesian surface associated with a Riemannian metric space. This Riemannian metric space is pictured by drawing the unit ball of the metric at some points of the domain. A link with differential geometry is proposed in [Alauzet 2011a].*

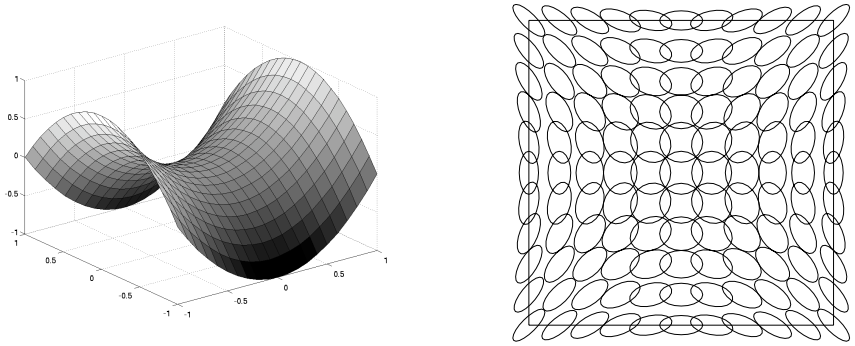


Figure 1.4: *Left, example of a Cartesian surface embedded in* $\mathbb{R}^3$*. Right, geometric visualization of a Riemannian metric space* $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in[0,1]\times[0,1]}$ *associated with this surface. At some points* $\mathbf{x}$ *of the domain, the unit ball of* $\mathcal{M}(\mathbf{x})$ *represented by ellipses is drawn.*

There is no global notion of scalar product in a Riemannian metric space, thus no global distance or norm. However, we can extend the notions of length, angle and volume from the Euclidian space case. To do so, we have to consider the variation of the metric in space. The following **geometrical quantities** are defined:

- the length of edge $\mathbf{e} = \overrightarrow{PQ}$ parametrized by $\gamma : t \in [0,1] \longmapsto P + t\,\overrightarrow{PQ}$ is computed with the following formula:

$$\ell_{\mathcal{M}}(\mathbf{e}) = \int_0^1 ||\gamma'(t)||_{\mathcal{M}(\gamma(t))}\,\mathrm{d}t = \int_0^1 \sqrt{\overrightarrow{PQ}^T\,\mathcal{M}(P + t\,\overrightarrow{PQ})\,\overrightarrow{PQ}}\,\mathrm{d}t\,, \tag{1.13}$$

- the angle between two vectors $\mathbf{u_1} = \overrightarrow{PQ_1}$ and $\mathbf{u_2} = \overrightarrow{PQ_2}$ is the unique real $\theta \in [0,\pi]$ such that:

$$\cos\theta = \frac{(\mathbf{u_1},\,\mathbf{u_2})_{\mathcal{M}(P)}}{||\mathbf{u_1}||_{\mathcal{M}(P)}\,||\mathbf{u_2}||_{\mathcal{M}(P)}}\,, \tag{1.14}$$

- the volume of element K with respect to $\mathbf{M}$ is more difficult to apprehend. Indeed, due to metric variations, element K, as seen with respect to metric field $\mathbf{M}$ is generally curved: it

is not a simplex anymore and normally, its volume should be computed with an integration formula:

$$|K|_{\mathcal{M}} = \int_K \sqrt{\det \mathcal{M}(\mathbf{x})}\, \mathrm{d}\mathbf{x}\,. \tag{1.15}$$

However, this volume can be approximated at first order:

$$|K|_{\mathcal{M}} \approx |K|_{\mathcal{I}} \sqrt{\det \mathcal{M}(G_K)}, \quad \text{where } G_K \text{ is the barycenter of } K. \tag{1.16}$$

1.1.2 Duality between discrete and continuous mesh : notion of unit mesh

This local relation of equivalence concerns elements, and needs to be somehow extended to whole meshes. Intuitively, the notion of Riemann metric space is going to play that role. The main difficulty is to take into account the variation of the function $\mathbf{x} \mapsto \mathcal{M}(\mathbf{x})$. The analysis can be simplified if $\mathbf{M}$ is rewritten as follows, separating its local and global properties. A Riemannian metric space $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is locally written :

$$\forall \mathbf{x} \in \Omega, \quad \mathcal{M}(\mathbf{x}) = d^{\frac{2}{3}}(\mathbf{x})\, \mathcal{R}(\mathbf{x}) \begin{pmatrix} r_1^{-\frac{2}{3}}(\mathbf{x}) & & \\ & r_2^{-\frac{2}{3}}(\mathbf{x}) & \\ & & r_3^{-\frac{2}{3}}(\mathbf{x}) \end{pmatrix} \mathcal{R}^T(\mathbf{x}), \tag{1.17}$$

where

- density d is equal to: $d = (\lambda_1\lambda_2\lambda_3)^{\frac{1}{2}} = (h_1h_2h_3)^{-1}$, with λ_i the eigenvalues of $\mathcal{M}$
- anisotropic quotients r_i are equal to: $r_i = h_i^3\,(h_1h_2h_3)^{-1}$
- $\mathcal{R}$ is the eigenvectors matrix of $\mathcal{M}$ representing the orientation.

The density d controls only the local level of accuracy of $\mathbf{M}$: (increasing or decreasing it does not change the anisotropic properties or the orientation), while the anisotropy is given by the anisotropic quotients and the orientation by matrix $\mathcal{R}$. The notion of complexity $\mathcal{C}$ of $\mathbf{M}$ can also be defined:

$$\mathcal{C}(\mathbf{M}) = \int_\Omega d(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_\Omega \sqrt{\det(\mathcal{M}(\mathbf{x}))}\, \mathrm{d}\mathbf{x}. \tag{1.18}$$

This quantifies the global level of accuracy of $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$.

From this formulation of a Riemannian metric field arises a duality between meshes and Riemannian metric spaces. This duality is justified by the strict analogy between the following discrete and continuous notions: orientation vs. $\mathcal{R}$, stretching vs. r_i, size vs. d and number of elements vs. $\mathcal{C}(\mathbf{M})$. However, the class of discrete meshes represented by $\mathbf{M}$ is complex to describe.

If the notion of unit element in the previous section was quite immediate, things are more complicated when it comes to define an **unit mesh**. *Stricto sensu*, an unit mesh is a mesh

whose all edges are unit with respect to the prescribed metric field. However, the existence of a mesh made of unit elements is not assured. For example, in $\mathbb{R}^3$, let's take the simplest case of $\mathcal{M}(P) = \mathcal{I}(P)$ for each point P, *i.e.* the canonical Euclidian space. It is well known that $\mathbb{R}^3$ cannot be filled with regular tetrahedra (that are unit with respect to the identity metric). So the constraint on the sizes of the edges has to be relaxed. But this can lead to meshes with very bad quality elements (flat elements). So we have to add a constraint on the volume of the elements, through a quality function.

For this reason, we have to introduce the notion of **quasi unit element**. A tetrahedron is said to be quasi unit with respect to a metric field $\mathcal{M}$ if its edges are close to unit, *i.e.* $\forall i,\ \ell_{\mathcal{M}}(\mathbf{e_i}) \in [\frac{1}{\sqrt{2}}, \sqrt{2}]$. To avoid elements with a null volume (see [Loseille 2011a]), we have to add a constraint on the volume, which is achieved through a quality function:

$$Q_{\mathcal{M}}(K) = \frac{\sqrt{3}}{216} \frac{\left(\sum_{i=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e_i})\right)^{\frac{3}{2}}}{|K|_{\mathcal{M}}} \in [1, +\infty]\,. \tag{1.19}$$

For the regular tetrahedron, the quality function is equal to 1, whereas it tends to $+\infty$ for a null volume tetrahedron. So an element close to a perfectly unit element has a quality close to 1. This leads to the following definition. A tetrahedron K defined by its list of edges $(\mathbf{e_i})_{i=1...6}$ is said **quasi-unit** for Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ if

$$\forall i \in [1,6], \quad \ell_{\mathcal{M}}(\mathbf{e_i}) \in \left[\frac{1}{\sqrt{2}}, \sqrt{2}\right] \quad \text{and} \quad Q_{\mathcal{M}}(K) \in [1, \alpha] \ \text{ with } \ \alpha > 1\,. \tag{1.20}$$

The definition of unit mesh consequently becomes : a **unit mesh** with respect to a Riemannian metric space $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is a mesh made of quasi-unit elements.

In practice, it is this definition that is used in meshing software. For any kind of desired mesh (uniform, adapted isotropic, adapted anisotropic), **the mesh generator will generate a mesh that is unit with respect to the prescribed Riemannian metric space**. Thus the resulting mesh is uniform and isotropic in the Riemannian metric space while it is adapted and anisotropic in the canonical Euclidian space. This is illustrated in Figure 1.5. This idea has turned out to be a huge breakthrough in the generation of anisotropic adapted meshes.

1.1.3 Duality between discrete and continuous interpolation error: notion of continuous linear interpolate

The model that we have just defined is used to obtain an analytic expression of the optimal mesh. The purpose now in this section is to obtain a still valid equivalence between a continuous error estimate for any function on any continuous mesh and a discrete error estimate on a discrete mesh.

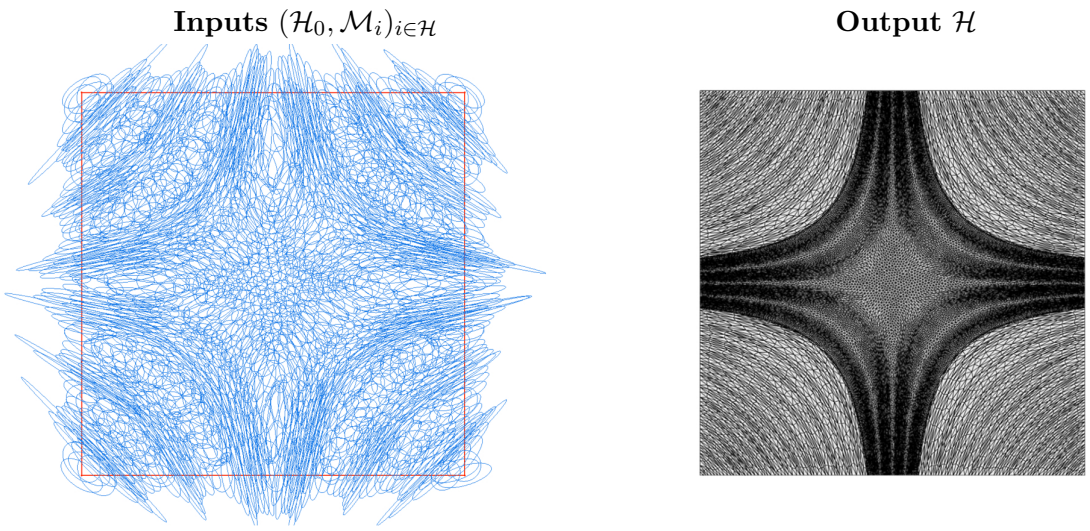


Figure 1.5: *Metric-based mesh generation. Left, specified Riemannian metric space. Right, unit mesh in the prescribed Riemannian metric space which becomes adapted anisotropic in the Euclidean space.*

In mathematical terms, let $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ be a continuous mesh of a domain Ω and let u be a smooth representation This function u is a non-linear function which is assumed to be twice continuously differentiable.

We define the following approximation space from the standard $\mathbb{P}_1$ FEM approximation space. Let $H^1(\Omega)$ be the Sobolev space defined as the set of applications of $L^2(\Omega)$ such that their first weak derivative exists and also belongs to $L^2(\Omega)$.

$$H^1(\Omega) = \{u \in \mathrm{L}^2(\Omega) \mid Du \in \mathrm{L}^2(\Omega)\}.$$

Let us introduce the following approximation space :

$$V_h = \{\varphi \in H^1(\Omega) \mid \varphi_{h|K} \text{ is affine } \forall K \in \mathcal{H}\}.$$

The usual $\mathbb{P}^1$-projector Π_h is defined such as $\Pi_h u$ is exact on each vertex of the element K :

$$\Pi_h : H^1(\Omega) \mapsto V_h \mid \Pi_h u(\mathbf{x}) = \varphi(\mathbf{x}), \forall \mathbf{x} \text{ vertex of } \mathcal{H}$$

In this section, we want to seek a well-posed definition of the continuous linear interpolation error $\|u - \pi_{\mathcal{M}} u\|_{L^1(\Omega)}$ related to a continuous mesh $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ which also implies a well-posed definition of a linear continuous interpolate $\pi_{\mathcal{M}} u$. And obviously, we would like the continuous linear interpolation error to be a reliable mathematical model of $\|u - \Pi_h u\|_{L^1(\Omega_h)}$.

In this sense, the interpolation error is first derived locally (on an element) for a quadratic function, then is extended to a global definition (for any point of the domain).

Local continuous interpolate

Let us first consider a quadratic function u defined on a domain $\Omega \subset \mathbb{R}^n$, and a continuous element $\mathcal{M}$. For all unit discrete elements K with respect to $\mathcal{M}$, the interpolation error of u in L^1 norm does not depend on the element shape and is only a function of the Hessian H_u of u and of continuous element $\mathcal{M}$. For more details see [Loseille 2011a].

- In 3D, for all unit elements K for $\mathcal{M}$, the following equality holds:

$$\|u - \Pi_h u\|_{L^1(K)} = \frac{\sqrt{2}}{240} \det(\mathcal{M}^{-\frac{1}{2}}) \operatorname{trace}(\mathcal{M}^{-\frac{1}{2}} H_u \mathcal{M}^{-\frac{1}{2}}). \tag{1.21}$$

- In 2D, for all unit elements K for $\mathcal{M}$, the following equality holds:

$$\|u - \Pi_h u\|_{L^1(K)} = \frac{\sqrt{3}}{64} \det(\mathcal{M}^{-\frac{1}{2}}) \operatorname{trace}(\mathcal{M}^{-\frac{1}{2}} H_u \mathcal{M}^{-\frac{1}{2}}).$$

For all the discrete elements that are unit with respect to $\mathcal{M}$, the interpolation error is the same, and is only based on continuous quantities (metric and Hessian). This last remark is important, since it shows metrics contain all the information required to compute the interpolation error, and are thus well adapted for anisotropic control of this error.

Global continuous interpolate

To define a global continuous linear interpolate, the problem is once again how to move from an expression valid for *one continuous element* to an expression for the case in which *the metric varies point-wise*. Let us now suppose that the continuous mesh $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is varying and that the function u is no more quadratic but only twice continuously differentiable. Equality (1.21) does not hold anymore, but all the terms of the right-hand side $\mathcal{M}$ and H are still well defined continuously.

Let's consider a point $\mathbf{a} \in \Omega$. In the vicinity of $\mathbf{a}$, we denote u_Q the quadratic approximation of smooth function u as the truncated second order Taylor expansion of u and $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ reduces to $\mathcal{M}(\mathbf{a})$ in the tangent space. There exists a unique function $\pi_{\mathcal{M}}$ such that:

$$\forall \mathbf{a} \in \Omega\,, \quad |u - \pi_{\mathcal{M}} u|(\mathbf{a}) \;=\; \frac{\|u_Q - \Pi_h u_Q\|_{L^1(K)}}{|K|} \;=\; \frac{1}{20} \operatorname{trace}\big(\mathcal{M}(\mathbf{a})^{-\frac{1}{2}} \,|H(\mathbf{a})|\, \mathcal{M}(\mathbf{a})^{-\frac{1}{2}}\big)\,, \tag{1.22}$$

for every K unit element with respect to $\mathcal{M}(\mathbf{a})$.

This result underlines another discrete-continuous duality by pointing out a continuous counterpart of the interpolation error. For this reason, the following formalism was adopted:

$\pi_{\mathcal{M}}$ is called *continuous linear interpolate* and $|u - \pi_{\mathcal{M}} u|$ represents the continuous dual of the interpolation error illustrated in cases of a concave or a convex function on Figure 1.6. From a practical point of view, we deduce the following analogy. Given an unit mesh $\mathcal{H}$ of a domain Ω_h with respect to a continuous mesh $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$, the global interpolation error is:

$$\|u - \Pi_h u\|_{L^1(\Omega_h)} = \sum_{K\in\mathcal{H}} \|u - \Pi_h u\|_{L^1(K)}. \tag{1.23}$$

In the continuous case, the discrete summation becomes an integral:

$$\|u - \pi_{\mathcal{M}} u\|_{L^1(\Omega)} = \int_\Omega |u - \pi_{\mathcal{M}} u|(\mathbf{x})\, \mathrm{d}\mathbf{x}. \tag{1.24}$$

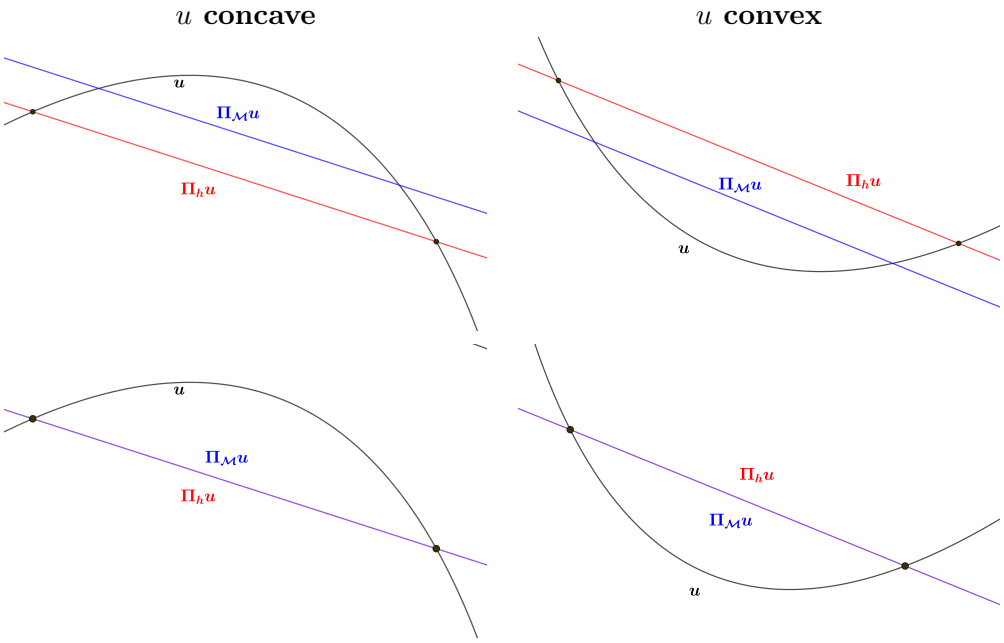


Figure 1.6: *Representations of discrete* (red) *and continuous* (blue) *projectors for u concave and u convex for an unspecified mesh* (first row) *and for an unit mesh* (second row).

There is no global guarantee on the reliability of the continuous interpolation error given by Relation (1.24), and in particular, there is no *a priori* relationship between (1.23) and (1.24). The only guarantee is the local equivalence given by Equation (1.22). However, the local guarantee becomes global when the mesh is unit with respect to a constant metric tensor and when the function is quadratic. In the latter case, by neglecting error due to the boundary discretization, we have the equality:

$$\|u - \Pi_h u\|_{L^1(\Omega_h)} = \|u - \pi_{\mathcal{M}} u\|_{L^1(\Omega)}\,, \tag{1.25}$$

for all unit meshes $\mathcal{H}$ with respect to $(\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$.

Several examples are given in [Loseille 2011b], both analytic and numerical, that confirm the validity of this analogy. They show that the model is accurate and the equivalence (1.23)$\approx$ (1.24) is observed even for non quadratic functions and non-constant continuous meshes, and that the error due to the fact that the mesh generator generates edges with length not strictly equal to one is negligible. In particular, the range for the lengths of the edges given in Section 1.1.2 ensures reliable numerical results.

1.1.4 Summary

We have presented a framework that draws a correspondence between the discrete domain and the continuous domain. This framework is summarized in Table 1.1.

1.2 Mesh adaptation

1.2.1 Steady mesh adaptation

Studying steady simulations is a necessary prerequisite for a better understanding of unsteady simulations. Let us first detail the feature-based mesh adaptation algorithm and the attached error analysis for steady problems then we turn to the description of goal-oriented mesh adaptation in the steady case.

Feature-based mesh adaptation for steady flows

The problem can be simplified to the specification of a mesh that is optimal for the interpolation error.

Mesh adaptation consists in finding the mesh that minimizes a certain error on a domain $\Omega \in \mathbb{R}^n$, for a certain sensor function u. The error we consider is the interpolation error, that we control in L^p norm - depending on the choice of p, different aspects of the solution are captured, as will be seen later. The problem is stated *a priori*:

$$\text{Find } \mathcal{H}_{L^p} \text{ having } N \text{ vertices such that } \mathbf{E}_{L^p}(\mathcal{H}_{L^p}) = \min_{\mathcal{H}} \|u - \Pi_h u\|_{L^p(\Omega_h)}. \qquad (P)$$

(P) is too complex to be solved directly, since the unknown, *i.e.* the mesh, is made up of vertices and their topology, which is far too many degrees of freedom. Besides, several optimal meshes can be found for one sensor function (think about merely swapping an element), so the problem is ill-posed. On the other hand, it is possible to show that the problem moved to the continuous domain is well posed, and can be solved using calculus of variations. What is more, the continuous formulation of the adaptation problem uses a global point of view, while usual methods focus on a local analysis of the error. The reformulated problem is:

DISCRETE	CONTINUOUS
Element K	Continuous element = Metric tensor $\mathcal{M}$
Element volume $\lvert K\rvert$	$d^{-1} = \sqrt{\det(\mathcal{M}^{-1})}$
Mesh $\mathcal{H}$ of Ω_h	Riemannian metric space $\mathbf{M}(\mathbf{x}) = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$
Number of vertices N_v	Complexity $\mathcal{C}(\mathcal{M}) = \displaystyle\int_\Omega d(\mathbf{x})\,\mathrm{d}\mathbf{x}$
$\mathbb{P}^1$ interpolate Π_h	$\mathbb{P}^1$-continuous interpolate $\pi_\mathcal{M}$
Local element-wise interpolation error $e_h(K) = \lvert\lvert u - \Pi_h u\rvert\rvert_K$ $e_h(K) = \dfrac{\lvert K\rvert}{40}\displaystyle\sum_{i=1}^{6} \mathbf{e}_i^T \lvert H_u\rvert \mathbf{e}_i$ (for u quadratic)	Local point-wise interpolation error $e(\mathbf{x}) = (u - \pi_\mathcal{M} u)(\mathbf{x})$ $e(\mathbf{x}) = \dfrac{1}{20}\,\mathrm{trace}\big(\mathcal{M}(\mathbf{x})^{-\frac{1}{2}} \lvert H(\mathbf{x})\rvert \mathcal{M}(\mathbf{x})^{-\frac{1}{2}}\big)$
Global continuous interpolation error $\displaystyle\sum_{K\in\mathcal{H}} \lvert\lvert u - \Pi_h u\rvert\rvert_K$	Global interpolation error $\displaystyle\int_{\mathbf{x}\in\Omega} e(\mathbf{x})\,\mathrm{d}\mathbf{x}$

Table 1.1: The continuous mesh model draws a correspondence between the discrete domain and the continuous domain.

$$\text{Find } \mathbf{M}_{L^p} \text{ having a complexity of } \mathcal{N} \text{ such that } \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \|u - \pi_\mathcal{M} u\|_{L^p(\Omega)}. \quad (1.26)$$

Using the definition of the linear continuous interpolate $\pi_\mathcal{M}$ given by Equation (1.22), the well-posed global optimization problem of finding the optimal continuous mesh minimizing the

continuous interpolation error in L^p norm can be established:

$$\begin{aligned} \text{Find } \mathbf{M}_{L^p} \text{ such that } \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) &= \min_{\mathbf{M}} \left(\int_\Omega \left(u(\mathbf{x}) - \pi_{\mathcal{M}} u(\mathbf{x}) \right)^p \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}} \\ &= \min_{\mathbf{M}} \left(\int_\Omega \mathrm{trace}\left(\mathcal{M}(\mathbf{x})^{-\frac{1}{2}} |H_u(\mathbf{x})| \mathcal{M}(\mathbf{x})^{-\frac{1}{2}} \right)^p \mathrm{d}\mathbf{x} \right)^{\frac{1}{p}}, \end{aligned} \tag{1.27}$$

under the constraint $\mathcal{C}(\mathbf{M}) = \displaystyle\int_\Omega d(\mathbf{x})\, \mathrm{d}\mathbf{x} = \mathcal{N}$. The constraint on the complexity notably avoids the trivial solution where all $(h_i)_{i=1,n}$ are zero which provides a null error.

Contrary to a discrete analysis, this problem can be solved globally by using a calculus of variations that is well-defined on the space of continuous meshes. In [Loseille 2011b], it is proved that Problem (1.27) admits a unique solution $\mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ which is locally defined by:

$$\mathcal{M}_{L^p}(\mathbf{x}) = \mathcal{N}^{\frac{2}{n}} \left(\int_\Omega \det(|H_u(\bar{\mathbf{x}})|)^{\frac{p}{2p+n}} \, \mathrm{d}\bar{\mathbf{x}} \right)^{-\frac{2}{n}} \det(|H_u(\mathbf{x})|)^{\frac{-1}{2p+n}} \, |H_u(\mathbf{x})| \,. \tag{1.28}$$

Moreover, it verifies the following properties:

- $\mathbf{M}_{L^p}$ is unique
- $\mathbf{M}_{L^p}$ is locally aligned with the eigenvectors basis of H_u and has the same anisotropic quotients as H_u
- $\mathbf{M}_{L^p}$ provides an optimal explicit bound of the interpolation error in L^p norm:

$$\|u - \pi_{\mathcal{M}_{L^p}} u\|_{L^p(\Omega)} = n\, \mathcal{N}^{-\frac{2}{n}} \left(\int_\Omega \det\left(|H_u|\right)^{\frac{p}{2p+n}} \right)^{\frac{2p+n}{np}}. \tag{1.29}$$

- For a sequence of continuous meshes having an increasing complexity with the same orientation and anisotropic quotients $(\mathbf{M}^{\mathcal{N}}_{L^p})_{\mathcal{N}=1\ldots\infty}$, the asymptotic order of convergence verifies:

$$\|u - \pi_{\mathcal{M}^N_{L^p}} u\|_{L^p(\Omega)} \leq \frac{Cst}{\mathcal{N}^{2/n}}. \tag{1.30}$$

Relation (1.30) points out a global second order of mesh convergence.

Anisotropic mesh adaptation is a non-linear problem, therefore an iterative procedure is required to solve this problem. For stationary simulations, an adaptive computation is carried out via a mesh adaptation loop inside which an algorithmic convergence of the mesh-solution couple is sought. This mesh adaptation loop is schematized in Algorithm 1 where $\mathcal{H}$, $\mathcal{S}$ and $\mathbf{M}$ denote respectively meshes, solutions and metric fields.

Algorithm 1 Feature-based mesh adaptation loop for steady flows

Input : Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted complexity $\mathcal{N}$

For $i = 0, n_{adap}$

1. $(\mathcal{S}_i)$= Compute solution with the flow solver from pair $(\mathcal{H}_i, \mathcal{S}_i^0)$;

   If $i = n_{adap}$ break;

2. $(\mathbf{M}_{L^p,i})$ = Compute metric $(\mathbf{M}_{L^p})$ according to selected error estimate from $(\mathcal{H}_i, \mathcal{S}_i)$;

3. $(\mathcal{H}_{i+1})$ = Generate a new adapted mesh form pair $(\mathcal{H}_i, \mathbf{M}_{L^p,i})$;

4. $(\mathcal{S}_{i+1}^0)$ = Interpolate new initial solution from $(\mathcal{H}_{i+1}, \mathcal{H}_i, \mathcal{S}_i)$;

End For

Before detailing the different steps of the numerical algorithm, let's focus on two major difficulties of it:

- The solution u of the problem is not known. We only can access to the numerical approximation u_h.
- As for all numerical simulations, a control of the approximation error $u - u_h$ is expected.

We demonstrate how this problem can be simplified to the specification of a mesh that is optimal for the interpolation error. We describe how the interpolation theory is applied when only u_h, a piecewise linear approximation of the solution, is known. Indeed, in this particular case, the interpolation error estimate is not applied directly to u nor to u_h.

Let V_h^1 be the space of continuous piecewise linear function associated with a given mesh $\mathcal{H}$ of domain Ω_h. We denote by R_h a reconstruction operator applied to numerical approximation u_h. This reconstruction operator can be either a recovery process [Zienkiewicz 1992], a hierarchical basis [Bank 1993], or an operator connected to an a posteriori estimate [Huang 2010a]. We assume that the reconstruction $R_h u_h$ is better than u_h for a given norm $\|.\|$ in the sense that:

$$\|u - R_h u_h\| \leq \alpha \|u - u_h\| \quad \text{where} \quad 0 \leq \alpha < 1\,.$$

From the triangle inequality, we deduce:

$$\|u - u_h\| \;\leq\; \frac{1}{1-\alpha}\|R_h u_h - u_h\|\,.$$

If reconstruction operator R_h has the property: $\Pi_h R_h \phi_h = \phi_h\,,\ \forall \phi_h \in V_h^1$, the approximation error of the solution can be bounded by the interpolation error of reconstructed function $R_h u_h$:

$$\|u - u_h\| \leq \frac{1}{1-\alpha}\|R_h u_h - \Pi_h R_h u_h\|\,.$$

From previous section, if $\mathcal{H}_{L^p}$ is an optimal mesh to control the interpolation error in L^p norm of $R_h u_h$, then the following upper bound of the approximation error can be exhibited:

$$\|u - u_h\|_{L^p(\Omega_h)} \leq \frac{n\mathcal{N}^{-\frac{2}{n}}}{1-\alpha} \left(\int_\Omega \det\left(|H_{R_h u_h}(\mathbf{x})|\right)^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x} \right)^{\frac{2p+n}{np}} .$$

Remark 2. *It is important to note that $\mathcal{M}_{L^p}$ defined by Relation (1.35) applied to $R_h u_h$ does not allow us to generate an optimal adapted mesh to control the approximation error $\|u - u_h\|$. If all assumptions are verified then we have an upper bound which allows that such generated adapted meshes controls the approximation error.*

Now, we details each step of the mesh adaptation algorithm.

Step 1 : The flow solver `Wolf` . All the numerical simulations of this thesis are concerned with the compressible Euler equations. Results on Navier-Stokes equation are not discussed in this thesis. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the 3D compressible Euler equations for mass, momentum and energy conservation read:

$$\frac{\partial W}{\partial t} + \nabla \cdot \mathcal{F}(W) = 0, \tag{1.31}$$

where W is the vector of conservative variables and $\mathcal{F}$ is the convection operator $\mathcal{F}(W) = (\mathcal{F}_1(W), \mathcal{F}_2(W), \mathcal{F}_3(W))$ with:

$$W = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \mathcal{F}_1(W) = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ (\rho E + p)u \end{pmatrix}, \mathcal{F}_2(W) = \begin{pmatrix} \rho v \\ \rho uv \\ \rho v^2 + p \\ \rho vw \\ (\rho E + p)v \end{pmatrix}, \mathcal{F}_3(W) = \begin{pmatrix} \rho w \\ \rho uw \\ \rho vw \\ \rho w^2 + p \\ (\rho E + p)w \end{pmatrix}.$$

We have denoted by ρ the density, $\mathbf{u} = (u, v, w)$ the Cartesian velocity vector, $E = T + \frac{\|\mathbf{u}\|^2}{2}$ the total energy and $p = (\gamma - 1)\rho T$ the pressure with $\gamma = 1.4$ the ratio of specific heats and T the temperature.

To discretize convective terms `Wolf` uses the mixed-element-volume (MEV) approach [Alauzet 2010a] initiated by Dervieux et al. in [Stoufflet 1987, Fezoui 1989, Debiez 2000, Cournède 2006]. It is a vertex-centered finite volume scheme applied to tetrahedral unstructured meshes. This scheme uses a particular edge-based formulation with upwind elements. For the following, it is important to specify the considered semi-discretization. The system of partial differential equations can be written in semi-discrete form for a vertex P_i as:

$$|C_i| \frac{dW_i}{dt} = R(W_i) ,$$

where R_i is the residual, *i.e.,* the discretization of the convective operator using an approximate Riemann solver and of the boundary conditions, and $|C_i|$ is the area/volume of the dual finite volume cell associated with P_i.

For an explicit time discretization, the semi-discretized system reads:

$$\frac{|C_i|}{\delta t_i^n}\left(W_i^{n+1} - W_i^n\right) \;=\; R(W_i^n)$$

where W^n is the state at iteration n and δt_i^n is the local time step at iteration n.

For implicit time discretization, we have :

$$\frac{|C_i|}{\delta t_i^n}\left(W_i^{n+1} - W_i^n\right) \;=\; R(W_i^{n+1})$$

which is linearized as:

$$\left(\frac{|C_i|}{\delta t_i^n} I_d - \frac{\partial R}{\partial W}(W_i^n)\right)\left(W_i^{n+1} - W_i^n\right) \;=\; R(W_i^n)$$

where $\dfrac{\partial R}{\partial W}(W_i^n)$ contributes the i^{th} line of the matrix. We then rewrite the linearized system in compact form for all vertices in the mesh as:

$$\mathbf{A}^n\,\delta\mathbf{W}^n = \mathbf{R}^n$$

$$\text{where} \quad \mathbf{A}^n = \frac{|C|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial W} \quad \text{and} \quad \delta\mathbf{W}^n = \mathbf{W}^{n+1} - \mathbf{W}^n\,.$$

Step 2 : The metric module `Metrix` computes the continuous mesh and performs the metric field gradation [Alauzet 2010b]. The optimal continuous mesh $\mathbf{M}_{L^p}(u)$ minimizing Problem (1.27) is given in (1.35). The metric gradation field is used to smooth and control the variation of the obtained optimal continuous mesh.

Step 3 : The local adaptive remesher `Feflo.a` generates an unit mesh with respect to a prescribed metric field. `Feflo.a` is a classical metric-based local remesher [Loseille 2017] based on a unique cavity operator which is a generalization of standard operators (insertion, collapse, swap of edges and faces, vertex smoothing) [Loseille 2013]. The cavity operator also performs combinations of standard operators in a very natural way. To this end, modifications on the cavity are done to either favor a modification, that would have been rejected with the standard operator, or to improve the final quality by combining automatically many standard operators at once. One important capability is to adapt the volume and the surface mesh in a coupled way so that a valid 3D mesh is always guaranteed on output.

Step 4 : The software component `Interpol` to project linearly the solution defined on the previous mesh onto the new mesh. The first step is to localize the new vertices in the element of the previous mesh. The second step consists in interpolating linearly the solution using the barycentric coordinates. More details on the interpolation stage can be obtained in [Alauzet 2010b, Alauzet 2016].

Goal-oriented mesh adaptation for steady flows

In the previous paragraph, metric-based anisotropic mesh adaptation method is limited to the minimization of some interpolation errors for some solution fields called sensors and did not take into account the PDE of the considered problem. If, for many applications, this simplifying standpoint is an advantage, there are also many applications where Hessian-based adaptation is far from optimal regarding the way the degrees of freedom are distributed in the computational domain. Indeed, Hessian-based methods aim at controlling the interpolation error but this purpose is not often so close to the objective that consists in obtaining the best solution of a PDE. This is particularly true in many engineering applications where a specific function needs to be accurately evaluated : lift, drag, heat flux, pressure field, etc. Hessian-based adaptation has not been especially designed to address this issue.

The objective in goal-oriented adaptation is to minimize the error on the considered functional j, details can be found in [Loseille 2010b] in the case of compressible Euler equations. For simplicity's sake boundary terms are not mentioned in the following. The approximation

Algorithm 2 Goal-oriented mesh adaptation loop for steady flows

Input : Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted complexity $\mathcal{N}$

For $i = 1, n_{adap}$

1. $(\mathcal{S}_i)$= Compute solution with the flow solver from pair $(\mathcal{H}_i, \mathcal{S}_i^0)$;
   If $i = 1$ break;
2. $(\mathcal{S}_i^*)$= Compute adjoint solution with the flow solver from $(\mathcal{H}_i, \mathcal{S}_i, j_i)$;
3. $(\mathbf{M}_{go,i})$ = Compute metric $(\mathbf{M}_{go})$ according to selected error estimate from $(\mathcal{H}_i, \mathcal{S}_i, \mathcal{S}_i^*)$;
4. $(\mathcal{H}_{i+1})$ = Generate a new adapted mesh form pair $(\mathcal{H}_i, \mathbf{M}_{go,i})$;
5. $(\mathcal{S}_{i+1}^0)$ = Interpolate new initial solution from $(\mathcal{H}_{i+1}, \mathcal{H}_i, \mathcal{S}_i)$;

End For

error on the functional $j(W)$ is given by the following error estimate:

$$|j(W) - j(W_h)| \leq \int_{\Omega_h} |\nabla W^*| \, |\mathcal{F}(W) - \Pi_h \mathcal{F}(W)| \mathrm{d}\Omega_h \tag{1.32}$$

where W^* is the adjoint state. We observe that this estimate is expressed in term of the interpolation error in L^1 norm of the Euler fluxes applied to the continuous solution W weighted by the gradient of the continuous adjoint state W^*.

The goal-oriented mesh adaptation loop is given by Algorithm 2. There is one more step to compute the adjoint state. Let us detail the steps 2 and 3 as all other steps listed earlier stay the same :

Step 2 : The adjoint flow solver `Wolf` . Let $j(W)$ be the considered output functional. Using the same notations as in the previous section, the adjoint state $\mathbf{W}^*$ is solution of the following linear system:

$$\mathbf{A}^* \, \mathbf{W}^* = \frac{\partial j}{\partial W}(\mathbf{W})$$

where $\mathbf{A}^*$ is the transpose of the jacobian matrix: $\mathbf{A}^* = \left(-\frac{\partial \mathbf{R}^n}{\partial W}\right)^T$.

Step 3 : The metric module `Metrix` computes the following continuous goal-oriented optimal metric and performs the metric field gradation. Using the same reasoning as for the error estimator for a sensor but applied to the goal-oriented error estimate (1.32), the error estimate is rewritten in a continuous form:

$$|j(W) - j(W_h)| \approx \mathbf{E}_{go}(\mathbf{M}) = \int_\Omega |\nabla W^*| \cdot |\mathcal{F}(W) - \pi_{\mathcal{M}} \mathcal{F}(W)| \, \mathrm{d}\Omega \, , \tag{1.33}$$

where $\mathbf{M} = (\mathcal{M}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is a continuous mesh and $\pi_{\mathcal{M}}$ is the continuous linear interpolate. We are now focusing on the following (continuous) mesh optimization problem:

$$\begin{aligned} \text{Find } \mathbf{M}_{go} \text{ such that } \mathbf{E}_{go}(\mathbf{M}_{go}) &= \min_{\mathbf{M}} \int_\Omega |\nabla W^*| \cdot |\mathcal{F}(W) - \pi_{\mathcal{M}} \mathcal{F}(W)| \, \mathrm{d}\Omega \\ &= \min_{\mathbf{M}} \int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x}) \, |\mathbf{H}_{go}(\mathbf{x})| \, \mathcal{M}^{-\frac{1}{2}}(\mathbf{x}) \right) \mathrm{d}\mathbf{x} \end{aligned}$$

under the constraint $\mathcal{C}(\mathbf{M}) = \mathcal{N}$ and where we have introduced the goal-oriented Hessian-metric given in 3D by

$$|\mathbf{H}_{go}(\mathbf{x})| = \sum_{j=1}^{5} \left(\left| \frac{\partial W_j^*}{\partial x}(\mathbf{x}) \right| \cdot \left| H(\mathcal{F}_1(W_j))(\mathbf{x}) \right| + \left| \frac{\partial W_j^*}{\partial y}(\mathbf{x}) \right| \cdot \left| H(\mathcal{F}_2(W_j))(\mathbf{x}) \right| + \left| \frac{\partial W_j^*}{\partial z}(\mathbf{x}) \right| \cdot \left| H(\mathcal{F}_3(W_j))(\mathbf{x}) \right| \right) . \tag{1.34}$$

The formulation of the optimal goal-oriented continuous mesh $\mathbf{M}_{go} = (\mathcal{M}_{go}(\mathbf{x}))_{\mathbf{x}\in\Omega}$ is locally given by:

$$\mathcal{M}_{go}(\mathbf{x}) = \mathcal{N}^{\frac{2}{n}} \left(\int_\Omega \det(|\mathbf{H}_{go}(\bar{\mathbf{x}})|)^{\frac{1}{2+n}} \, \mathrm{d}\bar{\mathbf{x}} \right)^{-\frac{2}{n}} \det(|\mathbf{H}_{go}(\mathbf{x})|)^{\frac{-1}{2+n}} \, |\mathbf{H}_{go}(\mathbf{x})| \, . \tag{1.35}$$

Note that we have the same expression as for the feature-based case in L^1 norm where the Hessian of the sensor has been replaced by the goal-oriented Hessian-metric.

Remark 3.

- *To minimize the function j we establish a continuous optimisation. But it's today impossible to solve this problem in a continuous way. To overcome this problem, we choose to discretize it. The nodal gradients of the state function and of the adjoint function are recovered from the gradients to the elements using a L^2 local projection and Clément's interpolation operator [Clément 1975].*
- *The details of the computation for goal-oriented mesh adaptation will be detailed in chapter 5.*
- *In this document, the metric optimization is defined as continuous and is always discretized to be solved.*

1.2.2 Unsteady mesh adaptation

Feature-based mesh adaptation for unsteady flows

In the context of time-dependent problems, the error analysis must control spatial and temporal error. In [Olivier 2011a, Alauzet 2016], time discretization errors are not taken into account but the focus was made on a space-time analysis of the spatial error. In other words, the work seeks for the optimal space-time mesh controlling the space-time spatial discretization error. For the type of considered simulations, the assumption is made that *as an explicit time scheme is used for time advancing, then the error in time is controlled by the error in space under CFL condition.* This has been demonstrated under specific conditions in [Alauzet 2007]. As long as this hypothesis holds, the spatial interpolation error provides a fair measure of the total space-time error of the discretized unsteady system. Notably in the case of implicit time advancing solvers, the analysis would have to be completed to take into account the temporal discretization error, as is done in [Coupez 2013], which defines the optimal time discretization (*i.e.,* time steps).

Our goal is to solve an unsteady PDE which is set in the computational space-time domain $\mathcal{Q} = \Omega \times [0,T]$ where T is the (positive) maximal time and $\Omega \subset \mathbb{R}^n$ is the spatial domain. Let Π_h be the usual P^1 projector, we extend it to time-dependent functions:

$$\left(\Pi_h \varphi\right)(t) = \Pi_h\left(\varphi(t)\right), \ \forall\, t \in \ [0,T]\,. \tag{1.36}$$

The considered problem of mesh adaptation consists in finding the space-time mesh $\mathcal{H}$ of $\Omega_h \times [0,T]$ that minimizes the space-time linear interpolation error $u - \Pi_h u$ in L^p norm. The problem is thus stated in an a priori way:

$$\text{Find } \mathcal{H}_{L^p} \text{ having } N_{st} \text{ vertices such that } \ \mathbf{E}_{L^p}(\mathcal{H}_{L^p}) = \min_{\mathcal{H}} \|u - \Pi_h u\|_{L^p(\Omega_h \times [0,T])}\,. \tag{1.37}$$

This problem is ill-posed and has far too many unknowns to be solved directly, as was explained previously. So it is rewritten in the continuous mesh framework under its continuous form:

$$\text{Find } \mathbf{M}_{L^p} = (\mathcal{M}_{L^p}(\mathbf{x},t))_{(\mathbf{x},t)\in\mathcal{Q}} \text{ such that } \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega\times[0,T])}\,, \quad (1.38)$$

under the space-time constraint:

$$\mathcal{C}_{st}(\mathbf{M}) = \int_0^T \tau(t)^{-1} \left(\int_\Omega d_{\mathcal{M}}(\mathbf{x},t)\ \mathrm{d}\mathbf{x} \right)\ \mathrm{d}t = \mathcal{N}_{st}\,, \quad (1.39)$$

where $\tau(t)$ is the time step used at time t of interval $[0,T]$. Introducing the continuous interpolation error, we recall that we can write the continuous error model as follows:

$$\mathbf{E}_{L^p}(\mathbf{M}) = \left(\int_0^T \int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t)|H_u(\mathbf{x},t)|\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) \right)^p \ \mathrm{d}\mathbf{x}\ \mathrm{d}t \right)^{\frac{1}{p}}\,, \quad (1.40)$$

where H_u is the Hessian of sensor u. To find the optimal space-time continuous mesh, Problem (3.3-3.4) is solved in two steps: first, a spatial minimization is done for a fixed t, then a temporal minimization is performed where the time step τ is specified by the user as a function of time $t \to \tau(t)$. It leads to the expression of the optimal space-time metric $\mathbf{M}_{L^p}$ for a prescribed time step $\tau(t)$ [Alauzet 2016]:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}_{st}^{\frac{2}{n}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+n}}\, \mathcal{K}(t)\mathrm{d}t \right)^{-\frac{2}{n}} \tau(t)^{\frac{2}{2p+n}} \left(\det|H_u(\mathbf{x},t)|\right)^{-\frac{1}{2p+n}} |H_u(\mathbf{x},t)|\,, \quad (1.41)$$

where $\mathcal{K}(t) = \left(\int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x} \right)$.

The computation of the optimal instantaneous continuous mesh given by Relation (3.12) involves a global normalization term which requires the knowledge of quantities over the whole simulation time frame. Thus, the complete simulation must be performed before evaluating any space-time continuous mesh. To solve this issue, we suggest to consider a global fixed-point mesh adaptation algorithm covering the whole time frame $[0,T]$. This iterative algorithm is used to converge the non-linear mesh adaptation problem, *i.e.,* converging the mesh-solution couple. This is also a way to predict the solution evolution and to adapt the mesh accordingly.

Moreover, the previous analysis provides the optimal size of the adapted meshes for each time level. Hence, this analysis requires the mesh to be adapted at each flow solver time step which is inconceivable in practical applications. We propose to use a coarse adapted discretization of the time axis. The basic idea consists in splitting the simulation time frame $[0,\ T]$ into n_{adap} adaptation sub-intervals:

$$[0,\ T] \ = \ [0 = t_0,\ t_1] \cup \ldots \cup [t_i,\ t_{i+1}] \cup \ldots \cup [t_{n_{adap}-1},\ t_{n_{adap}} = T]\,,$$

and to keep the same adapted spatial mesh for each time sub-interval. On each sub-interval, the mesh is adapted to control the solution accuracy from t_i to t_{i+1}. Consequently, the time-dependent simulation is performed with n_{adap} different adapted meshes. This drastically reduces the number of remeshings during the simulation, hence the number of solution transfers. The unsteady mesh adaptation algorithm is presented in Algorithm 3.

We now present the modified or new steps involve for time-accurate mesh adaptation.

Step 1 (a) : The software component `Interpol` to project the solution defined on the previous mesh onto the new mesh between each sub-interval. This stage becomes crucial in the context of unsteady problems as error due to the interpolation step may accumulate throughout the simulation. To minimize the error and to be consistant with the considered PDE (equations of conservation), a P^1-exact conservative interpolation is considered. More details on the conservative interpolation stage can be obtained in [Alauzet 2010c, Alauzet 2016].

Step 1 (b) : The flow solver `Wolf` . The main difference between the steady and the unsteady case is that a global time stepping is considered for time-dependent problems. If we consider an explicit first-order time integration scheme, then the semi-discrete unsteady model

Algorithm 3 Feature-based mesh adaptation for unsteady flows

Input : Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity $\mathcal{N}_{st}$

 # Fixed-point loop to converge the global space-time mesh adaptation problem

For $j = 1, n_{ptfx}$

 # Adaptive loop to advance the solution in time on time frame $[0, T]$

1. **For** $i = 0, n_{adap}$
   (a) $(\mathcal{S}_{0,i}^j)$= Interpolate conservatively next sub-interval initial solution from pair $(\mathcal{H}_{i-1}^j, \mathcal{S}_{i-1}^j, \mathcal{H}_i^j)$;
   (b) $(\mathcal{S}_i^j)$= Compute solution with the flow solver on sub-interval from pair $(\mathcal{H}_i^j, \mathcal{S}_i^0)$;
   (c) $|\mathbf{H}_{L^1}|_i^j$ = Compute sub-interval Hessian-metric from solution sample $(\mathcal{H}_i^j, \{\mathcal{S}_i^j(k)\}_{k=1}^{nk})$;

   EndFor
2. $\mathcal{C}^j$ = Compute space-time complexity from all Hessian metrics $\{|\mathbf{H}_{L^1}|_i^j\}_{i=1}^{n_{adap}}$
3. $\{\mathbf{M}_{L^p,i}^j\}_{i=1}^{n_{adap}}$ = Compute all sub-interval metrics according to error estimate from $(\mathcal{C}^j, \{|\mathbf{H}_{L^1}|_i^j\}_{i=1}^{n_{adap}})$;
4. $\{\mathcal{H}_i^{j+1}\}_{i=1}^{n_{adap}}$ = Generate all sub-interval adapted meshes from pair $(\{\mathcal{H}_i^j, \mathbf{M}_{L^p,i}^j\}_{i=1}^{n_{adap}})$;

End For

at time t^n is:

$$|C_i|\frac{W_i^n - W_i^{n-1}}{\delta t^n} = R(W_i^{n-1})\,.$$

Step 3 : The metric module `Metrix` . We extend the previous analysis to the fixed-point mesh adaptation algorithm context - described in Algorithm 3 - where the simulation time interval [0,T] is split into n_{adap} *sub-intervals* $[t_{i-1}, t_i]$ for $i = 1, .., n_{adap}$. Each spatial mesh $\mathbf{M}^i$ is then kept constant during each sub-interval $[t_{i-1}, t_i]$. We could consider this partition as a *time discretization of the mesh adaptation problem.* Following the previous section analysis, we deduce the optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}}$ defined locally by:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{n}} \left(\sum_{j=1}^{n_{adap}} \mathcal{K}^j \left(\int_{t_{j-1}}^{t_j} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+n}} \right)^{-\frac{2}{n}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{-\frac{2}{2p+n}} (\det \mathbf{H}^i_{L^1}(\mathbf{x}))^{-\frac{1}{2p+n}}\ \mathbf{H}^i_{L^1}(\mathbf{x}) \tag{1.42}$$

and the associated optimal space-time error:

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = n\,\mathcal{N}_{st}^{-\frac{2}{n}} \left(\sum_{i=1}^{n_{adap}} \mathcal{K}^i \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+n}} \right)^{\frac{2p+n}{np}}, \tag{1.43}$$

where $\mathcal{K}^i = \left(\int_\Omega \left(\det \mathbf{H}^i_{L^1}(\mathbf{x}) \right)^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x} \right)$ and the Hessian-metric $\mathbf{H}^i_{L^1}$ on sub-interval i is the time-average of the Hessian of the sensor u:

$$\mathbf{H}^i_{L^1}(\mathbf{x}) = \int_{t_{i-1}}^{t_i} |H_u(\mathbf{x}, t)|\, \mathrm{d}t\,. \tag{1.44}$$

Remark 4. *The mesh adaptation with unsteady fixed-point method does not converge to the 2nd order for singularities because of the uniform time step. Multi-rate methods as in [Itam 2017] can be a solution to overcome the problem. An other issue is to modify he number of sub-intervals during the adaptive loop [Alauzet 2016].*

Goal-oriented mesh adaptation for unsteady flows

We have seen the "in-house" goal-oriented mesh adaptation technique for steady problems and the feature-based mesh adaptation for unsteady flows. Now, let's turn to the goal-oriented mesh adaptation for unsteady flows.

The objective in goal-oriented adaptation for unsteady problems is to minimize the error on the functional $j(W)$ which is now integrated in time. An *a priori* error analysis [Belme 2012] gives the following error estimate for the compressible Euler equations where we have again neglected the boundary terms for the sake of simplicity :

$$|j(W) - j(W_h)| = \int_0^T \int_\Omega \left| \frac{\partial W^*}{\partial t} \right| |W - \Pi_h W| \mathrm{d}\Omega \mathrm{d}t \ + \ \int_0^T \int_\Omega |\nabla W^*| \cdot |\mathcal{F}(W) - \Pi_h \mathcal{F}(W)| \mathrm{d}\Omega \mathrm{d}t \tag{1.45}$$

The Algorithm 4 is further complicated by the fact that the unsteady adjoint solver must be computed backward in time (eg. step 2) after computing the solution over the simulation time frame.

Algorithm 4 Goal-Oriented Mesh Adaptation for Unsteady Flows

Input : Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity $\mathcal{N}_{st}$

 # Fixed-point loop to converge the global space-time goal-oriented mesh adaptation problem

For $j = 1, n_{ptfx}$

 # Adaptive loop to compute forward the solution state in time on time frame $[0, T]$

1. **For** $i = 1, n_{adap}$

   (a) $\mathcal{S}_{0,i}^j$ = Interpolate conservatively next sub-interval initial solution from $(\mathcal{H}_{i-1}^j, \mathcal{S}_{i-1}^j, \mathcal{H}_i^j)$;

   (b) $\mathcal{S}_i^j$ = Compute solution on sub-interval from pair $(\mathcal{H}_i^j, \mathcal{S}_{0,i}^j)$;

   EndFor

 # Adaptive loop to compute backward the adjoint state in time on time frame $[T, 0]$

2. **For** $i = n_{adap}, 1$

   (a) $(\mathcal{S}^*)_i^j$ = Interpolate previous sub-interval final adjoint state from $(\mathcal{H}_i^j, \mathcal{H}_{i+1}^j, (\mathcal{S}_0^*)_{i+1}^j)$;

   (b) $(\mathcal{S}_0^*)_i^j$ = Compute backward adjoint state on sub-interval from $(\mathcal{H}_i^j, \mathcal{S}_i^j, (\mathcal{S}^*)_i^j)$;

   (c) $|\mathbf{H}_{go,L^1}|_i^j$ = Compute sub-interval goal-oriented Hessian-metric from sample $(\mathcal{H}_i^j, \{\mathcal{S}_i^j(k), (\mathcal{S}^*)_i^j(k)\}_{k=1}^{nk})$;

   EndFor

3. $\mathcal{C}^j$ = Compute space-time complexity from all goal-oriented Hessian-metrics $\{|\mathbf{H}_{go,L^1}|_i^j\}_{i=1}^{n_{adap}}$;

4. $\{\mathcal{M}_i^j\}_{i=1}^{n_{adap}}$ = Compute all sub-interval unsteady metrics $(\mathcal{C}^j, \{|\mathbf{H}_{go,L^1}|_i^j\}_{i=1}^{n_{adap}})$;

5. $\{\mathcal{H}_i^{j+1}\}_{i=1}^{n_{adap}}$ = Generate all sub-interval adapted meshes $(\{\mathcal{H}_i^j,\ \mathcal{M}_i^j\}_{i=1}^{n_{adap}})$;

EndFor

Now, we describe the new steps of the unsteady goal-oriented mesh adaptation algorithm which are the computation of the unsteady adjoint state and the unsteady goal-oriented error estimate.

Step 2 : The unsteady adjoint flow solver `Wolf` . Let $j(W)$ be the considered output functional. using the same notations as previously, the unsteady adjoint state $\mathbf{W}^*$ is solution of:

$$-\frac{\partial \mathbf{W}^*}{\partial t} + \mathbf{A}^*\,\mathbf{W}^* = \frac{\partial j}{\partial W}(\mathbf{W})$$

where $\mathbf{A}^*$ is the transpose of the jacobian matrix: $\mathbf{A}^* = \left(-\frac{\partial \mathbf{R}^n}{\partial W}\right)^T$. If we consider an explicit first-order time integration scheme, then the semi-discrete unsteady adjoint model at time t^n

is:

$$|C_i|\frac{W_i^{*,n-1}-W_i^{*,n}}{-\delta t^n}=\frac{\partial j}{\partial W}(W_i^{n-1})+\left(W_i^{*,n}\right)^T\frac{\partial R}{\partial W}(W_i^{n-1})\,.$$

We notice that the computation of the unsteady adjoint state at time t^{n-1} requires the knowledge of the adjoint state at time t^n and the solution state at time t^{n-1}. Therefore, the solution state must be known for the whole simulation time frame $[0,T]$ to be able to compute the adjoint state.

Step 4 : The metric module `Metrix` . Following the analysis for the steady goal-oriented mesh adaptation and for the unsteady mesh adaptation, the continuous mesh adaptation problem to solve is:

$$\begin{aligned}\text{Find } \mathbf{M}_{go} \text{ such that } \mathbf{E}_{go}(\mathbf{M}_{go}) &= \min_{\mathbf{M}}\int_0^T\int_\Omega\left(\left|\frac{\partial W^*}{\partial t}\right||W-\pi_{\mathcal{M}}W|+|\nabla W^*|\cdot|\mathcal{F}(W)-\pi_{\mathcal{M}}\mathcal{F}(W)|\right)\mathrm{d}\Omega\mathrm{d}t\\ &= \min_{\mathbf{M}}\int_0^T\int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t)\,|\mathbf{H}_{go}(\mathbf{x},t)|\,\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t)\right)\mathrm{d}\mathbf{x}\mathrm{d}t\end{aligned}$$

under the space-time constraint:

$$\mathcal{C}_{st}(\mathbf{M})=\int_0^T\tau(t)^{-1}\left(\int_\Omega d_{\mathcal{M}}(\mathbf{x},t)\;\mathrm{d}\mathbf{x}\right)\;\mathrm{d}t=\mathcal{N}_{st}\,,$$

where $\tau(t)$ is the time step used at time t of interval $[0,T]$. The optimal continuous mesh is defined on the space time domain $\mathcal{Q}=\Omega\times[0,T]$: $\mathbf{M}_{go}=(\mathcal{M}_{go}(\mathbf{x},t))_{(\mathbf{x},t)\in\mathcal{Q}}$ and the unsteady goal-oriented Hessian-metric $\mathbf{H}_{go}$ in 3D is

$$\begin{aligned}|\mathbf{H}_{go}(\mathbf{x},t)| &= \sum_{j=1}^{5}\left(\left|\frac{\partial W_j^*}{\partial t}(\mathbf{x},t)\right|\cdot\left|H(W_j)(\mathbf{x},t)\right|+\left|\frac{\partial W_j^*}{\partial x}(\mathbf{x},t)\right|\cdot\left|H(\mathcal{F}_1(W_j))(\mathbf{x},t)\right|\right.\\ &+ \left.\left|\frac{\partial W_j^*}{\partial y}(\mathbf{x},t)\right|\cdot\left|H(\mathcal{F}_2(W_j))(\mathbf{x},t)\right|+\left|\frac{\partial W_j^*}{\partial z}(\mathbf{x},t)\right|\cdot\left|H(\mathcal{F}_3(W_j))(\mathbf{x},t)\right|\right).\end{aligned}$$

As for the feature-based unsteady mesh adaptation, solving the above optimization problem will provide the optimal instantaneous adapted mesh for each time level. Again, we split the simulation time frame into n_{adap} sub-intervals and we derive an optimal spatial mesh for each sub-interval. The same space-time error analysis provides the expression of the optimal unsteady goal-oriented mesh for each sub-interval $\mathbf{M}_{go}=\left\{\mathbf{M}_{go}^i\right\}_{i=1,..,n_{adap}}$ defined locally by:

$$\mathcal{M}_{go}^i(\mathbf{x})=\mathcal{N}_{st}^{\frac{2}{n}}\left(\sum_{j=1}^{n_{adap}}\mathcal{K}^j\left(\int_{t_{j-1}}^{t_j}\tau(t)^{-1}\mathrm{d}t\right)^{\frac{2p}{2p+n}}\right)^{-\frac{2}{n}}\left(\int_{t_{i-1}}^{t_i}\tau(t)^{-1}\mathrm{d}t\right)^{-\frac{2}{2p+n}}(\det\mathbf{H}_{go,L^1}^i(\mathbf{x}))^{-\frac{1}{2p+n}}\;\mathbf{H}_{go,L^1}^i(\mathbf{x})\tag{1.46}$$

where $\mathcal{K}^i=\left(\int_\Omega\left(\det\mathbf{H}_{go,L^1}^i(\mathbf{x})\right)^{\frac{p}{2p+n}}\mathrm{d}\mathbf{x}\right)$ and the Hessian-metric $\mathbf{H}_{go,L^1}^i$ on sub-interval i is the time-average of the unsteady goal-oriented Hessian-metric:

$$\mathbf{H}_{go,L^1}^i(\mathbf{x})=\int_{t_{i-1}}^{t_i}|\mathbf{H}_{go}(\mathbf{x},t)|\,\mathrm{d}t\,.\tag{1.47}$$

As for the steady case, we notice that we have the same expression as for the feature-based case where the Hessian of the sensor has been replaced by the unsteady goal-oriented Hessian-metric. The associated optimal space-time error is:

$$\mathbf{E}_{go}(\mathbf{M}_{go}) = n\,\mathcal{N}_{st}^{-\frac{2}{n}} \left(\sum_{i=1}^{n_{adap}} \mathcal{K}^i \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+n}} \right)^{\frac{2p+n}{np}} . \tag{1.48}$$

1.3 Mesh adaptation for moving geometries

Given that feature-based and goal-oriented anisotropic mesh adaptations have proved their efficiency to reduce the CPU time of steady and unsteady simulations while improving their accuracy it has become evident to extend the theory to time-dependent problems with moving geometries. Yet this task is far from straightforward. In [Barral 2015] time-accurate anisotropic mesh adaptation for 3D time-dependent problems involving body-fitted moving geometries has been implemented. It has only considered feature-based error estimate. Let us present this work in this section.

1.3.1 Mesh adaptation problem

The feature-based unsteady mesh adaptation cannot directly be extended to moving mesh simulations because it does not take into account the movement of the mesh. Indeed, the error analysis from which the optimal continuous mesh is deduced, assumes that the mesh is constant in time inside a sub-interval, therefore it does not take into account the local deformation of the mesh which necessarily impacts local errors. As regards the discrete representation of the metric field for moving meshes, two options are available. The first approach is an Eulerian approach: a metric is associated with a fixed position in space, thus the metric field is a "background" field evolving in space and time but independently from the moving mesh. The second approach is a Lagrangian approach, which is considered in this work, in that case a metric is attached to a moving vertex and moves with it, *i.e.,* we have $\mathcal{M}(\mathbf{x}(t), t)$ which we will write $\mathcal{M}(\mathbf{x}(t))$ in what follows.

In the context of dynamic meshes, the optimization problem is still given by Relations (3.3) and (3.4) of the unsteady featured-based mesh adaptation problem. But it is not obvious that for dynamic meshes problem, the space-time error model is given by Relation (1.40).

To complete our reflection, the ALE framework is modelled as follow (see Figure 1.7). The simulation time frame $[0, T]$ is split into n_{adap} adaptation time steps (in green in the Figure 1.7) :

$$[0, T] = \bigcup_{i=0}^{n_{adap}} [t^i, t^{i+1}]$$

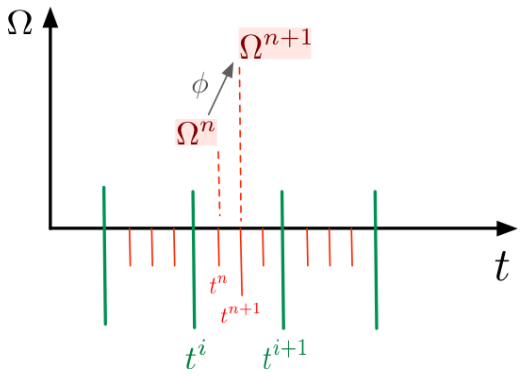


Figure 1.7: *Notations and mapping in ALE framework.*

As for the fixed-mesh case, to perform the space-time L^p error analysis with time sub-intervals, we first do a spatial minimization for a given sub-interval. We consider the i^{th} sub-interval $[t^i, t^{i+1}]$. Given the continuous mesh spatial complexity $\mathcal{N}^i$, we seek for the optimal Arbitrary-Lagrangian-Eulerian (ALE) continuous mesh $\mathbf{M}^{i,\mathrm{ALE}}_{L^p} = \left(\mathcal{M}^{i,\mathrm{ALE}}_{L^p}(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ for the whole sub-interval which is solution of the following problem:

$$\mathbf{E}^{i,\mathrm{ALE}}_{L^p}(\mathbf{M}^{i,\mathrm{ALE}}_{L^p}) = \min_{\mathbf{M}^i} \int_{t^i}^{t^{i+1}} \left(\int_{\Omega(t)} \mathrm{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t)) \, |H_u(\mathbf{x}(t),t)| \, (\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t)) \right)^p \mathrm{d}\mathbf{x}(t) \right) \mathrm{d}t$$

such that $\mathcal{C}(\mathbf{M}^i) = \mathcal{N}^i$. Note that the continuous mesh spatial complexity is constant on this sub-interval. We name it optimal ALE continuous mesh to stress that the continuous mesh is dynamic (*i.e.,* it evolves in time), it represents a moving adapted mesh and it is coupled with an ALE flow solver. And this is the heart of the problem : $\mathbf{M}^i$ has a dependence in time ! Thus we cannot move the integral over time into the trace to make the time-averaged Hessian-metric appear like in the fixed mesh analysis. The spatial minimization for a given sub-interval can not be done directly.

To solve this issue while preserving the adaptation of the mesh despite the mesh deformation, we are now looking for a mesh at the beginning t^i of each sub-interval $[t^i, t^{i+1}]$ that will be adapted to the solution at each time-step of this sub-interval once moved with the prescribed mesh displacement. The idea which is explained in the next paragraph is to exhibit the optimal instantaneous ALE continuous mesh minimizing the interpolation error in L^p norm at a given time $t \in [t^i, t^{i+1}]$. We shortly call it the ALE metric field because it takes into account the mesh deformation. It is then possible to map this continuous moving mesh of $\Omega(t),\ t \in [t^i, t^{i+1}]$, onto a continuous mesh of $\Omega(t^i)$. Thus, the error model can be written only on $\Omega(t^i)$ and the spatial minimization can be performed.

1.3.2 Optimal instantaneous ALE continuous mesh minimizing the interpolation error in L^p norm: the ALE metric

The computational domain is time-dependent $\Omega(t) \subset \mathbb{R}^3$ with $t \in [0, T]$. The problem for one time-step is the following. Let t^n and t^{n+1} (in red in the Figure 1.7) be two different times. And let $\Omega^n = \Omega(t^n)$ and $\Omega^{n+1} = \Omega(t^{n+1})$ be the spatial domains at t^n and t^{n+1} respectively. Let's note that generally $\Omega^n \neq \Omega^{n+1}$. We denote by $\mathbf{d}$ the given mesh displacement field and $\mathbf{x}^n = \mathbf{x}(t^n)$. Then, we want to find the optimal continuous mesh $\mathbf{M}_{L^p}^{n,\mathrm{ALE}} = (\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n))_{\mathbf{x}^n \in \Omega^n}$ defined on Ω^n from which we will generate a mesh at time t^n that, once moved with displacement $\mathbf{d}$, will be adapted to a sensor u^{n+1} at time t^{n+1} on domain Ω^{n+1}. We consider that u^{n+1} is a scalar sensor function, the extension to vector sensor functions being straightforward. The resolution of this problem leads to the optimal instantaneous ALE continuous mesh defined on Ω^n minimizing the interpolation error of sensor u^{n+1} in L^p norm at time t^{n+1} on Ω^{n+1} after being deformed by displacement $\mathbf{d}$. This ALE metric field involves the gradient of the mesh transformation between t^n and t^{n+1} to take into account the mesh deformation.

To give the result of the analysis involving spatial derivatives, it is very important to state on which domain/mesh these derivatives are computed. To this end, the following notations will be used:

- ∇^n denotes the gradient operator performed on domain Ω^n, *i.e.,* computed on mesh $\mathcal{H}^n$
- H^{n+1} denotes the Hessian operator performed on domain Ω^{n+1}, *i.e.,* computed on mesh $\mathcal{H}^{n+1}$. As operator H^{n+1} is always applied to sensor u^{n+1} at time t^{n+1}, simplified notation H_u^{n+1} will stand for $H^{n+1}[u^{n+1}]$
- $\mathcal{M}_{L^p}^{n+1}[u^{n+1}]$ denotes the point-wise optimal L^p metric for sensor u^{n+1} computed on domain Ω^{n+1}, *i.e.,* on mesh $\mathcal{H}^{n+1}$. Again, simplified notation $\mathcal{M}_{L^p}^{n+1}$ will stand for $\mathcal{M}_{L^p}^{n+1}[u^{n+1}]$
- $\mathbf{M}_{L^p}^{n+1} = (\mathcal{M}_{L^p}^{n+1}(\mathbf{x}))_{\mathbf{x} \in \Omega^{n+1}}$ is the associated continuous mesh with complexity $\mathcal{C}(\mathbf{M}_{L^p}^{n+1}) = \mathcal{N}^{n+1}$.

Now, let us introduce $\boldsymbol{\phi}$ the mapping between domains Ω^n and Ω^{n+1}:

$$\begin{array}{rccl} \boldsymbol{\phi}: & \Omega^n & \longrightarrow & \Omega^{n+1} \\ & \mathbf{x}^n & \longmapsto & \mathbf{x}^{n+1} = \boldsymbol{\phi}\left(\mathbf{x}^n\right), \end{array} \tag{1.49}$$

and $\mathbf{d}$ the corresponding mesh displacement field, such that:

$$\mathbf{x}^{n+1} = \boldsymbol{\phi}(\mathbf{x}^n) = \mathbf{x}^n + \mathbf{d}(\mathbf{x}^n). \tag{1.50}$$

Since $\boldsymbol{\phi}$ is a diffeomorphism, we have, for any infinitesimal vector $\delta\mathbf{x}^n \in \Omega^n$:

$$\delta\mathbf{x}^{n+1} = \left[\nabla^n \boldsymbol{\phi}(\mathbf{x}^n)\right]^T \delta\mathbf{x}^n \qquad \text{with} \qquad \nabla^n \boldsymbol{\phi}(\mathbf{x}^n) = \mathcal{I} + \nabla^n \mathbf{d}(\mathbf{x}^n), \quad \forall \mathbf{x}^n \in \Omega^n. \tag{1.51}$$

Finally, a $\widehat{.}$ operator is defined which transports a quantity from Ω^{n+1} to Ω^n. We note $\widehat{H_u^{n+1}}$ the Hessian of u^{n+1} computed on Ω^{n+1} and transported on domain Ω^n. This mathematically writes:

$$\begin{array}{rccl} \widehat{H_u^{n+1}}: & \Omega^n & \longrightarrow & \mathbb{R} \\ & \mathbf{x}^n & \longmapsto & H_u^{n+1}(\boldsymbol{\phi}(\mathbf{x}^n)) = \widehat{H_u^{n+1}}(\mathbf{x}^n)\,. \end{array}$$

To give the expression of the optimal continuous mesh, the demonstration of the result is given in [N. Barral 2017], we introduce the ALE Hessian-metric:

$$|H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)| = \left|\det \nabla^n \boldsymbol{\phi}(\mathbf{x}^n)\right|^{\frac{1}{p}} \left(\nabla^n \boldsymbol{\phi}(\mathbf{x}^n)\ |\widehat{H_u^{n+1}}(\mathbf{x}^n)|\ \left[\nabla^n \boldsymbol{\phi}(\mathbf{x}^n)\right]^T\right), \tag{1.52}$$

and the expression of the optimal continuous mesh in 3D defined on $\Omega(t^n)$ is:

$$\mathcal{M}_{L^p}^{n,\mathrm{ALE}}(\mathbf{x}^n) = \left(\mathcal{N}^{n+1}\right)^{\frac{2}{3}} \left(\int_{\Omega^n} \left(\det |H_u^{n,\mathrm{ALE}}(\bar{\mathbf{x}}^n)|\right)^{\frac{p}{2p+3}} \mathrm{d}\bar{\mathbf{x}}^n\right)^{-\frac{2}{3}} \left(\det |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)|\right)^{-\frac{1}{2p+3}} |H_u^{n,\mathrm{ALE}}(\mathbf{x}^n)|\,, \tag{1.53}$$

We find that we have the same expression as the optimal mesh in the stationary case (cf. Relation (1.35)) the absolute value of the Hessian of the sensor is replaced by the ALE Hessian-metric of the sensor.

1.3.3 Space-time error analysis for dynamic meshes

Section 1.3.2 provides the optimal instantaneous ALE continuous mesh which takes into account the mesh deformation. Now, as precised previously, we can extend the space-time error analysis with time sub-intervals done for fixed meshes to the case of dynamic meshes. The simulation time interval is split into n_{adap} sub-intervals. On each sub-interval, the mesh size (number of vertices) remains constant, but the mesh is deformed to follow the geometry displacement. At each time-step of the sub-interval, we want the moved mesh to be adapted to the current sensor. The key idea to perform the error analysis is to seek for the optimal dynamic continuous mesh at the beginning of the sub-interval, this continuous mesh being optimal for the whole sub-interval when deformed, instead of seeking for the expression of the optimal continuous mesh at each instant, *i.e.,* as a function of the time.

To perform the spatial minimization for a sub-interval, we consider the i^{th} sub-interval $[t^i, t^{i+1}]$. Given the continuous mesh spatial complexity $\mathcal{N}^i$, we seek for the optimal ALE continuous mesh $\mathbf{M}_{L^p}^{i,\mathrm{ALE}} = \left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ for the whole sub-interval which is solution of the following problem:

$$\mathbf{E}_{L^p}^{i,\mathrm{ALE}}(\mathbf{M}_{L^p}^{i,\mathrm{ALE}}) = \min_{\mathbf{M}^i} \int_{t^i}^{t^{i+1}} \left(\int_{\Omega(t)} \mathrm{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t))\,|H_u(\mathbf{x}(t),t)|\,(\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x}(t))\right)^p \mathrm{d}\mathbf{x}(t)\right) \mathrm{d}t\,. \tag{1.54}$$

such that $\mathcal{C}(\mathbf{M}^i) = \mathcal{N}^i$. The continuous mesh spatial complexity is constant on this sub-interval. To remove the time dependency of continuous mesh $\mathbf{M}^i$, we use the optimal instantaneous ALE continuous mesh. Indeed, continuous mesh $\left(\mathcal{M}^i(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ can be mapped back to $\Omega(t^i)$ using $\left(\mathcal{M}^{i,\mathrm{ALE}}(\mathbf{x}(t^i))\right)_{\mathbf{x}\in\Omega(t^i)}$ where $\boldsymbol{\phi}$ maps $\Omega(t^i)$ onto $\Omega(t)$. The interpolation error for each time t can be re-written at time t^i, see [N. Barral 2017]. As we seek for the dynamic continuous mesh at time t^i which is optimal to control the interpolation error for the whole sub-interval, we can recast Error Model (1.54) into the following error model where the metric is independent of the time:

$$\mathbf{E}^{i,\mathrm{ALE}}_{L^p}(\mathbf{M}^i) = \int_{t^i}^{t^{i+1}} \left(\int_{\Omega(t^i)} \mathrm{trace}\left((\mathcal{M}^{i,\mathrm{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \, |H^{i,\mathrm{ALE}}_u(\mathbf{x}(t^i))| \, (\mathcal{M}^{i,\mathrm{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \right)^p \mathrm{d}\mathbf{x}(t^i) \right) \mathrm{d}t \,,$$

where the $\mathbf{x}(t^i)$ are in domain $\Omega(t^i)$. The time dependency in $H^{i,\mathrm{ALE}}_u$ is hidden in mapping $\boldsymbol{\phi}$ and operator $\widehat{\cdot}$. The previous expression can now be written on $\Omega(t^i)$:

$$\mathbf{E}^{i,\mathrm{ALE}}_{L^p}(\mathbf{M}^i) = \int_{\Omega(t^i)} \mathrm{trace}\left((\mathcal{M}^{i,\mathrm{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \, \mathbf{H}^{i,\mathrm{ALE}}_u(\mathbf{x}(t^i)) \, (\mathcal{M}^{i,\mathrm{ALE}})^{-\frac{1}{2}}(\mathbf{x}(t^i)) \right)^p \mathrm{d}\mathbf{x}(t^i) \,,$$

where the time-average ALE Hessian-metric is: $\mathbf{H}^{i,\mathrm{ALE}}_u(\mathbf{x}(t^i)) = \displaystyle\int_{t^i}^{t^{i+1}} |H^{i,\mathrm{ALE}}_u(\mathbf{x}(t^i))| \, \mathrm{d}t$. The expression of the error has exactly the same form as in Problem (1.40), thus the spatial minimization gives the same optimal metric where $\mathbf{H}^i_u$ is replaced by $\mathbf{H}^{i,\mathrm{ALE}}_u$. Then, the temporal minimization leads to the following optimal space-time ALE continuous mesh $\mathbf{M}^{\mathrm{ALE}}_{L^p} = \{\mathbf{M}^{i,\mathrm{ALE}}_{L^p}\}_{i=1,..,n_{adap}}$:

$$\begin{aligned}
\mathcal{M}^{i,\mathrm{ALE}}_{L^p}(\mathbf{x}(t^i)) \;=\; & \mathcal{N}_{st}^{\frac{2}{3}} \left(\sum_{j=1}^{n_{adap}} \mathcal{K}^{j,\mathrm{ALE}} \left(\int_{t^j}^{t^{j+1}} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+3}} \right)^{-\frac{2}{3}} \\
& \left(\int_{t^i}^{t^{i+1}} \tau(t)^{-1} \mathrm{d}t \right)^{-\frac{2}{2p+3}} (\det \mathbf{H}^{i,\mathrm{ALE}}_u(\mathbf{x}(t^i)))^{-\frac{1}{2p+3}} \, \mathbf{H}^{i,\mathrm{ALE}}_u(\mathbf{x}(t^i)) \,,
\end{aligned}$$

where the $\mathbf{x}(t^i)$ are in domain $\Omega(t^i)$. The ALE continuous mesh dependence in time is hidden in $\mathbf{H}^{i,\mathrm{ALE}}_u$ by means of mapping $\boldsymbol{\phi}$. This way, the mesh generated at time t^i is adapted to the solution at any time $t > t^i$ within the sub-interval $[t^i, t^{i+1}]$ once moved with the mesh deformation displacement. We stress that preserving the number of degrees of freedom when moving the mesh is essential in this analysis.

1.3.4 Mesh adaptation algorithm

The optimal space-time ALE continuous mesh is designed to fit in the global fixed-point unsteady mesh adaptation algorithm described in Algorithm 3. Nevertheless, a few things have been modified to extend this algorithm to moving mesh ALE simulations. In fact, first, geometries move so mesh must be moved with all the difficulties that this implies to preserve accuracy and a good mesh quality all along the movement. All of this will be detailed more precisely in this thesis.

1.4 Conclusion

This first chapter has provided the basic wide background required by this thesis. It relies on the continuous mesh framework to exhibit an analytical expression of the optimal adapted mesh. We observe that for the steady and the unsteady cases we have always the same expression of the optimal mesh, the difference in the error estimates resides in the choice of the Hessian-metric which is simply the Hessian of the sensor for the simplest case (eg. steady feature-based mesh adaptation) or more complex combination of Hessian-metric for the other cases.

All the improvements achieved over the theories presented here will be mentioned throughout the dissertation as well as the newest developments regarding goal-oriented adaptation for moving meshes and adjoint solver computations for moving geometries. To this end, the main difficulties of this work are the extension of the unsteady adjoint solver in the context of the moving geometries and the goal-oriented error estimate in this context.

The main contributions of this thesis are:

- the definition, the implementation and the validation of implicit time integration scheme for the ALE flow solver (Chapter 2),
- the completion with respect to several steps for the unsteady mesh adaptation for moving geometries which complement thesis [Barral 2015] (Chapter 3). In particular, the importance to update the ALE metric in the flow solver to govern the mesh optimizations during the moving mesh algorithm,
- the development of the unsteady adjoint solver in the ALE framework which requires a backward in time moving mesh algorithm consistent with the forward in time one (Chapter 4)
- the time accurate goal-oriented mesh adaptation for moving geometries and its associated error estimate (Chapter 5).

Chapter 2

Resolution of Euler equations in the ALE framework

Fluid-structure interaction (FSI) simulations are unavoidable for a wide variety of subjects. They include simulation of aircraft flutters, ship propellers efficiency [Compère 2010], wind turbines efficiency [Bazilevs 2011], 2D airbag deployment and balloon inflation [Saksono 2007], releasing of a missile [Murman 2003, Hassan 2007]. In the acoustic field one can cite tuning forks [Froehle 2014]. And in the field of biology, simulation of aortic valves [Astorino 2009], cardiovascular systems [Formaggia 2009, Gerbeau 2014] or the simulation of jellyfish [Etienne 2010] also use FSI simulation. In the case of fluid-structure interaction, the framework is a movable solid whether stiff or deformable and a fluid whether gaseous or liquid. But other problems such as icing [Tong 2014, Pendenza 2014] or blast studies [Baum 1996] do not involve FSI but share a lot of common problems with FSI simulations. And all of these applications require appropriate computational methodologies customized to the specific context.

Three main types of methods are possible to treat the deformation of the domain : the multiphasic method, the mesh deformation method or the fictional domain method. The multiphasic approach initially refers to a multiphase flow then any fluid flow consisting of more than one phase or component. It is limited in the case of FSI simulation to the cases where the solid and the fluid can be described by the same equations with variable physical parameters assigned to each phase and advected during the movement of the interface. In general, the Lagrangian approach for a fluid simulation is conceivable and gives good results for some multiphase issues for example in the industry of nuclear energy [Causin 2005], but when the flow becomes complex, the interfaces displacement induces the deformation of the mesh and imposes a remeshing of the complete domain. In a realistic state, this recall occurs in very short times. An Eulerian formulation is so better suited to the simulation of a flow fluid. On the other hand, in the context of the fluid-structure interaction, a formulation purely eulerian does not allow to effectively monitor the phenomena at the fluid-solid interface and poses the question of the law of behavior to be imposed in the cells cut by this interface.
The mesh deformation approach with an Arbitrary Lagrangian Eulerian (ALE) approach is

built to avoid these drawbacks. This method allows to ensure a smooth transition between these two methods of modelling because the flow is calculated on a domain that is deformed to follow the movement of the interface (Lagrangian close to the solid) the rate of deformation does not follow that of the interior field. To our knowledge, this technique was introduced in the 70s in [Hirt 1974, Hughes 1981, Donea 1982]. Since then, so many developments have been made in that field that a complete list of them would not fit in this thesis. However, one may in particular refer to [Nkonga 1994, Baum 1994, Farhat 2001, Formaggia 2004, Mavriplis 2006, Hassan 2007, Hay 2014], which mainly focus on improving temporal schemes for ALE simulations. When the displacement of the geometry is small enough, slightly deforming the original mesh [Batina 1990, Degand 2002] can generally be acceptable. However, difficulties arise when the displacement of the structure is too important: the mesh quickly becomes distorted and the numerical error due to this distortion quickly becomes too great, until the elements of the mesh finally become invalid, and the simulation has to be stopped. This is the major difficulty in this method. The computational domain of the fluid varies depending on the time according to the displacement of the solid. And even if the great development of computing capacities has made it possible to run increasingly complex simulations, the engineers are still far from performing such simulations, largely due to the difficulty of handling the moving meshes induced by large deformations of the moving geometries. Specific strategies need to be developed to deal with the displacement of moving boundary problems. For now, this method is interesting for large displacements problems if the fluid domain is quite often remeshed or coupled with mesh adaptation.
It's to get around the question of remeshing that the fictional domain methods have been developed also called embedded methods. These embedded boundary approach [Bruchon 2009, Löhner 2001] uses meshes that are not body-fitted at all : the bodies are embedded in a fixed grid, and techniques such as level-sets are used to recover their moving boundaries. The inconvenient of this method especially is the low CFL condition for unstructured vertex-centered method.

All three approaches have their own strengths and weaknesses. The chosen strategy in this thesis to handle moving geometries is the body-fitted framework where the geometry is explicitly represented inside the mesh. This choice is motivated by its compliance with anisotropic mesh adaptation and by the fact that body-fitted approaches are the more accurate methods to simulate viscous flows (even if only inviscid flow will be considered in this thesis). For body-fitted moving mesh simulations, the whole mesh must be deformed to follow the moving boundaries. And for this strategy to be successful, it is essential to take into account this move into the numerical scheme. The ALE framework is the chosen strategy in this work and it has been studied using explicit time integration in [N. Barral 2017]. However, the use of an

explicit time integration schemes leads to an important restriction of the admissible time step because it is specified by the size of small elements (usually the smallest element) in the mesh. Explicit schemes are widely used for blast problems (violent flows), LES, aeroacoustic, ... to control efficiently dissipation and dispersion errors that can spoil the overall solution accuracy [J. Berland 2007, Löhner 2001, N. Gourdain 2009].

To reduce the time step restriction due to the CFL condition, improved time advancing schemes exist such as multi-rate approaches [Constantinescu 2007, B. Seny 2014].However, the efficiency these approaches is highly reduced when coupled with mesh adaptation because they increase the time steps in regions where mesh adaptation generates a coarse mesh.

The other strategy to reduce the time step restriction due to the CFL condition is to use implicit time integration schemes that are theoretically unconditionally stable (in practice, implicit schemes have a larger region of stability compared to explicit schemes, leading to larger time steps during time marching). These schemes are very attractive for quasi static flows, aeronautical smooth flow simulations (vortex shedding, ...), viscous flows, ... But, to see a gain in CPU time, the considered implicit time-step should be sufficiently large to overcome the over cost of the implicit time integration without impacting the solution accuracy.

Section 2.1 presents a reminder of moving mesh strategy [Alauzet 2014]. Then, the ALE flow solver `Wolf` is described, Section 2.2. Section 2.2.3 presents the different implicit time integration approaches that have been developed in this thesis. And finally, comparaison between explicit and implicit time integration schemes are provided in the section 2.4.

2.1 Mesh-connectivity-change moving mesh strategy

To handle moving boundaries, we adopt a body-fitted approach, with a single mesh: the inner vertices of the mesh are moved following the moving boundaries to preserve the validity of mesh (*i.e.* to prevent the mesh from getting tangled). Our strategy involves two main parts:

- Computing the mesh deformation: inner vertices are assigned a trajectory depending on the displacement of the boundaries, and thus a position for future time steps.
- Optimizing the mesh: the trajectories computed in the mesh deformation phase are corrected, and the connectivity of the mesh is modified to preserve the quality of the mesh.

This strategy, recalled below, has proven to be very powerful in 3D [Alauzet 2014], since large displacement of complex geometries can be performed while preserving a good mesh quality without any global remeshing (*i.e. without ever generating a whole new mesh*).

To our knowledge, very few examples of ALE solvers coupled with connectivity-change moving mesh techniques can be found in the literature. In [Kucharik 2008] a conservative interpolation is proposed to handle the swaps. In [Guardone 2011, Olivier 2011b] an ALE formulation of

the swap operator is built. However, these studies are limited to 2D. In [Barral 2015] a method for computations in 3D for moving geometries coupled with connectivity-change is done and the numerical solver used is an explicit time integration method.

2.1.1 Linear elasticity mesh deformation method

During the mesh deformation step, a displacement field is computed for the whole computational domain, given the displacement of its boundaries. Trajectories can thus be assigned to inner vertices, or in other words, positions at a future solver time step. Several techniques can be found to compute this displacement field:

- implicit or direct interpolation [de Boer 2007, Luke 2012], in which the displacement of the inner vertices is a weighted average of the displacement of the boundary vertices,
- solving PDEs - the most common of which being Laplacian smoothing [Löhner 1998], a spring analogy [Degand 2002] and a linear elasticity analogy [Baker 1999].

It is this last method that we selected, due to its robustness in 3D [Yang 2007]. The computational domain is assimilated to a soft elastic material, which is deformed by the displacement of its boundaries.

The inner vertices movement is obtained by solving an *elasticity-like equation* with a $\mathbb{P}_1$ Finite Element Method (FEM):

$$\operatorname{div}(\sigma(\mathcal{E})) = 0\,, \quad \text{with} \quad \mathcal{E} = \frac{\nabla \mathbf{d} + {}^T\nabla \mathbf{d}}{2}\,, \tag{2.1}$$

where σ and $\mathcal{E}$ are respectively the Cauchy stress and strain tensors, and $\mathbf{d}$ is the Lagrangian displacement of the vertices. The Cauchy stress tensor follows Hooke's law for isotropic homogeneous medium, where ν is the Poisson ratio, E the Young modulus of the material and λ, μ are the Lamé coefficients:

$$\sigma(\mathcal{E}) = \lambda \operatorname{trace}(\mathcal{E})\, \mathcal{I}_d + 2\,\mu\, \mathcal{E} \qquad \text{or} \qquad \mathcal{E}(\sigma) = \frac{1+\nu}{E}\sigma - \frac{\nu}{E} \operatorname{trace}(\sigma)\, \mathcal{I}_d\,.$$

In our context, ν is typically chosen of the order of 0.48, which corresponds to a nearly incompressible material [1]. Dirichlet boundary conditions are used and the displacement of vertices located on the domain boundary is strongly enforced in the linear system. The linear system is solved by a Conjugate Gradient algorithm coupled with an LU-SGS pre-conditioner. An advantage of elasticity-like methods is the opportunity they offer to adapt the local material properties of the mesh, especially its stiffness, according to the distortion and efforts borne by each element. In particular, the stiffness of the elements is increased for small elements, in order to limit their distortion. More details can be found in [Alauzet 2014].

[1] Note that the closer to 0.5 ν is, the harder to solve the system is.

2.1.2 Improving mesh deformation algorithm efficiency

The computation of the mesh deformation - here the solution of a linear elasticity problem - is known to be an expensive part of dynamic mesh simulations, and the fact that it is usually performed at every solver time step makes it all the more so.

We propose to combine several techniques to improve the time efficiency of this step. Some regions are rigidified, more specifically a few layers around tiny complex details of the moving bodies, with very small elements. They are moved with exactly the same rigid displacement as the corresponding body, thus avoiding very stiff elements in the elasticity matrix. On the other hand, the elasticity can be solved only on a reduced region, if the domain is big compared to the displacement. A coarse mesh can also be used to solve the elasticity problem, the displacement of the vertices then being interpolated on the computational mesh.

The major improvement we proposed is to reduce the number of mesh deformation computations: the elasticity problem is solved for a large time frame of length Δt instead of doing it at each solver time step δt. While there is a risk of a less effective mesh displacement solution, it is a worthwhile strategy if our methodology is able to handle large displacements while preserving the mesh quality. Solving the previously described mesh deformation problem once for large time frame could be problematic in the case of: (i) curved trajectories of the boundary vertices and (ii) accelerating bodies. To enhance the mesh deformation prescription, accelerated-velocity curved, *i.e.,* high-order, vertex trajectories are computed.

The paths of inner vertices can be improved if a constant acceleration $\mathbf{a}$ is provided to each vertex in addition to its speed, which results in an accelerated and curved trajectory. During time frame $[t, t+\Delta t]$, the position and the velocity of a vertex are updated as follows:

$$\begin{aligned} \mathbf{x}(t+\delta t) &= \mathbf{x}(t) + \delta t\, \mathbf{v}(t) + \frac{\delta t^2}{2}\mathbf{a} \\ \mathbf{v}(t+\delta t) &= \mathbf{v}(t) + \delta t\, \mathbf{a}\,. \end{aligned}$$

Prescribing a velocity vector and an acceleration vector to each vertex requires solving two elasticity systems. For both systems, the same matrix, thus the same pre-conditioner, is considered. Only boundary conditions change. If inner vertex displacement is sought for time frame $[t, t+\Delta t]$, boundary conditions are imposed by the location of the body at time $t+\Delta t/2$ and $t+\Delta t$. These locations are computed using body velocity and acceleration. Note that solving the second linear system is cheaper than solving the first one as a good prediction of the expected solution can be obtained from the solution of the first linear system. Now, to define the trajectory of each vertex, the velocity and acceleration are deduced from evaluated middle and final positions:

$$\begin{aligned} \Delta t\, \mathbf{v}(t) &= -3\,\mathbf{x}(t) + 4\,\mathbf{x}(t+\Delta t/2) - \mathbf{x}(t+\Delta t) \\ \frac{\Delta t^2}{2}\mathbf{a} &= 2\,\mathbf{x}(t) - 4\,\mathbf{x}(t+\Delta t/2) + 2\mathbf{x}(t+\Delta t)\quad . \end{aligned}$$

In this context, it is mandatory to make sure that the mesh remains valid for the whole time frame $[t, t + \Delta t]$, which is done by computing the sign of the volume of the elements all along their path [Alauzet 2014].

2.1.3 Local mesh optimization

In order to preserve the mesh quality between two mesh deformation computations, it has been proposed [Alauzet 2014] to couple mesh deformation with local mesh optimization using smoothing and generalized swapping to efficiently achieve large displacement in moving mesh applications. Connectivity changes are really effective in handling shear and removing highly skewed elements. Here, we briefly recall the mesh optimization procedure.

Mesh optimizations are performed regularly to preserve the quality of the mesh. The optimization procedure consists in one or several passes of vertex smoothing and one pass of generalized swapping. For 3D meshes, the quality of an element is measured in terms of the element shape by the quality function:

$$Q(K) = \frac{\sqrt{3}}{216} \frac{\left(\sum_{i=1}^{6} \ell^2_{\mathcal{M}}(\mathbf{e}_i) \right)^{\frac{3}{2}}}{|K|_{\mathcal{M}}} \quad \in [1, +\infty] \ , \tag{2.2}$$

where $\ell_{\mathcal{M}}(\mathbf{e})$ and $|K|_{\mathcal{M}}$ are edge length and element volume in metric $\mathcal{M}$. $Q(K) = 1$ corresponds to a perfectly regular element and $Q(K) < 2$ corresponds to excellent quality elements, while a high value of $Q(K)$ indicates a nearly degenerated element. For non-adapted meshes, the identity matrix $\mathcal{I}_3$ is chosen as metric tensor.

Mesh smoothing. The first mesh optimization tool is vertex smoothing which consists in relocating each vertex inside its ball of elements, *i.e.,* the set of elements having P_i as their vertex. For each tetrahedron K_j of the ball of P_i, a new optimal position P_j^{opt} for P_i can be proposed to form a regular tetrahedron:

$$P_j^{opt} = G_j + \sqrt{\frac{2}{3}} \frac{\mathbf{n}_j}{\ell(\mathbf{n}_j)} \, ,$$

where F_j is the face of K_j opposite vertex P_i, G_j is the center of gravity of F_j, $\mathbf{n}_j$ is the inward normal to F_j and $\ell(\mathbf{n}_j)$ the length of $\mathbf{n}_j$. The final optimal position P_i^{opt} is computed as a weighted average of all these optimal positions $\{P_j^{opt}\}_{K_j \supset P_i}$, the weight coefficients being the quality of K_j. This way, an element of the ball is all the more dominant if its quality in the original mesh is bad. Finally, the new position is analyzed: if it improves the worst quality of the ball, the vertex is directly moved to its new position.

Generalized edge/face swapping. The second mesh optimization tool to improve mesh quality is generalized swapping/local-reconnection. Let α and β be the two tetrahedra vertices opposite the common face $P_1P_2P_3$. Face swapping consists of suppressing this face and creating the edge $\mathbf{e} = \alpha\beta$. In this case, the two original tetrahedra are deleted and three new tetrahedra are created. This swap is called $2 \to 3$. The reverse operator can also be defined by deleting three tetrahedra sharing such a common edge $\alpha\beta$ and creating two new tetrahedra sharing face $P_1P_2P_3$. This swap is called $3 \to 2$.

A generalization of this operation exists and acts on shells of tetrahedra [Alauzet 2014, Frey 2008]. For an internal edge $\mathbf{e} = \alpha\beta$, the shell of $\mathbf{e}$ is the set of tetrahedra having $\mathbf{e}$ as common edge. The different edge swaps are generally denoted $n \to m$ where n is the size of the shell and m is the number of new tetrahedra. In this work, edge swaps $3 \to 2$, $4 \to 4$, $5 \to 6$, $6 \to 8$ and $7 \to 10$ have been implemented. In our algorithm, swaps are only performed if they improve the quality of the mesh.

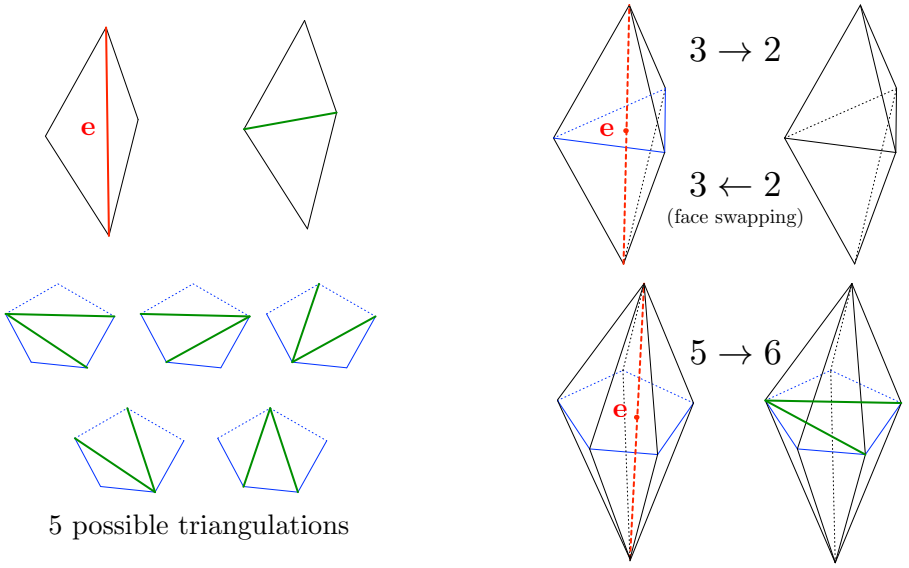


Figure 2.1: *Top left, the swap operation in two dimensions. Top right, edge swap of type* $3 \to 2$ *and face swap* $2 \to 3$*. Bottom left, the five possible triangulations of the pseudo-polygon for a shell having five elements. Bottom right, an example of* $5 \to 6$ *edge swap. For all these figures, shells are in black, old edges are in red, new edges in green and the pseudo-polygon is in blue.*

These operations are well-know in the field of mesh generation [J. F. Thompson 1999, Frey 2008], but are not necessarily efficient in the moving mesh context. Notably, performing too many of them results in slow codes, whereas the use of bad quality functions results in poor quality meshes. The interest of the method used here lies in how and when optimizations are performed. The mesh optimizations are performed entity by entity, and only when they

are needed. Smoothing is performed for every vertex above a prescribed quality threshold (the quality of a vertex is the maximal element quality of the elements in its ball[2]), provided the new position increases the quality of the worst element in its ball. Swaps are analyzed element by element if and only if the element quality is above a prescribed quality threshold. Then, swaps are performed if the quality of the considered elements decrease after the connectivity change. To avoid multiple identical checks, tetrahedra are treated in quality order from the worst to the best one. In the moving mesh context, the optimization operation is performed only if it verifies quality criteria on the current position of the mesh and on the final position given by the mesh deformation. This avoid to perform a mesh optimization which will degrade the quality of the mesh in the future. A key to performing efficient swaps in the moving mesh context is to allow a slight quality degradation in the future. Details on this optimization step can be found in [Alauzet 2014].

2.1.4 Handling of boundaries

The mesh of the boundaries is moved rigidly, and the vertices are not generally moved on the surface (no displacement in the tangential directions). However, in some cases, such as when a body is moving very close to the bounding box of the domain, it can be useful to move the vertices of the bounding box as well. In this case, we can allow tangential displacement on the boundary. For curved boundary, vertices are re-projected on the surface after the displacement. To do so, the displacements along the tangential axes are simply considered as new degrees of freedom. For instance, for a plane (x,y), the displacements along the x-axis and the y-axis are considered as degrees of freedom and are added to the elasticity system. The displacement along the z-axis is still set to 0, and thus is not added to the system. This can be generalized for any tangent plane.

2.1.5 Moving mesh algorithm

The overall connectivity-change moving mesh algorithm is described in Algorithm 5, where the different phases described above are put together. When coupled with a flow solver (see Sections 2.2), the flow solver is called after the optimization phase. In this algorithm, $\mathcal{H}$ stands for meshes, $\mathcal{S}$ for solutions, Q for quality (see Relation (2.2)), $\mathbf{d}_{|\partial\Omega_h}$ for the displacement on the boundary, and $\mathbf{v}$ and $\mathbf{a}$ for speed and acceleration. Δt and δt are time steps whose meaning is detailed below.

[2]The elements ball of a vertex is the set of elements sharing that vertex.

Algorithm 5 Connectivity-Change Moving Mesh Algorithm with Curved Trajectories

Input: $\mathcal{H}^0$, $\mathcal{S}^0$, Δt^0

While $(t < T^{end})$

1. $\Delta t = \Delta t^0$

2. Solve mesh deformation: compute vertices trajectories

   (a) $\left\{\mathbf{d}^{body}_{|\partial\Omega_h}(t + \Delta t/2)\right\}$ = Compute body vertex displacement from current translation speed $\mathbf{v}^{body}$, rotation speed $\boldsymbol{\theta}^{body}$ and acceleration $\mathbf{a}^{body}$ for $[t, t + \Delta t/2]$
   $\mathbf{d}(t + \Delta t/2)$ = Solve elasticity system $\left(\mathbf{d}^{body}_{|\partial\Omega_h}(t + \Delta t/2),\, \Delta t/2\right)$

   (b) $\left\{\mathbf{d}^{body}_{|\partial\Omega_h}(t + \Delta t)\right\}$ = Compute body vertex displacement from current translation speed $\mathbf{v}^{body}$, rotation speed $\boldsymbol{\theta}^{body}$ and acceleration $\mathbf{a}^{body}$ for $[t, t + \Delta t]$
   $\mathbf{d}(t + \Delta t)$ = Solve elasticity system $\left(\mathbf{d}^{body}_{|\partial\Omega_h}(t + \Delta t),\, \Delta t\right)$

   (c) $\{\mathbf{v}, \mathbf{a}\}$ = Deduce inner vertex speed and acceleration from both displacements $\{\mathbf{d}(t + \Delta t/2), \mathbf{d}(t + \Delta t)\}$

   (d) If predicted mesh motion is invalid then $\Delta t = \Delta t/2$ and goto 2.
   Else $T^{els} = t + \Delta t$

3. Moving mesh stage, with mesh optimizations and solver solution
   While $(t < T^{els})$

   (a) δt^{opt} = Get optimization time step $\left(\mathcal{H}^k, \mathbf{v},\, CFL^{geom}\right)$

   (b) δt^{solver} = Get solver time step $\left(\mathcal{H}^k, \mathcal{S}^k, \mathbf{v},\, CFL\right)$

   (c) $\delta t = \min(\delta t^{opt}, \delta t^{solver})$

   (d) If $t > t^{opt}$

      i. $\mathcal{H}^k$ = Swaps optimization $\left(\mathcal{H}^k, Q^{swap}_{target}\right)$

      ii. $\mathbf{v}^{opt}$ = Vertex smoothing $\left(\mathcal{H}^k, Q^{smoothing}_{target},\, Q_{max}\right)$

      iii. $t^{opt} = t + \delta t^{opt}$

   (e) $\mathcal{S}^{k+1}$ = Solve Equation of State $\left(\mathcal{H}^k, \mathcal{S}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}\right)$

   (f) $\mathcal{H}^{k+1}$ = Move the mesh and update vertex speed $\left(\mathcal{H}^k, \delta t, \mathbf{v}, \mathbf{v}^{opt}, \mathbf{a}\right)$

   (g) Check mesh quality. If too distorted: solve new deformation problem or stop.

   (h) $t = t + \delta t$

   EndWhile

EndWhile

In this algorithm, three time steps appear: a large one Δt for the mesh deformation computation, a smaller one δt^{opt} corresponding to the steps where the mesh is optimized, and the solver time step δt^{solver}. The first one, Δt, is currently set manually at the beginning of the computation. After each mesh deformation solution, the quality of the mesh in the future is analyzed: if the quality is too low, the mesh deformation is problem is solved again with a smaller Δt (*cf* step 2.(d)). Moreover, if the mesh quality degrades, a new mesh deformation solution is computed (*cf* step 3.(g)). The second one is computed automatically, using the CFL^{geom} parameter as described below. Determining the third one will be discussed in Section 2.2. If the solver time step is greater than the optimization time step, then the solver time step is truncated to follow the optimizations. If the solver time step is smaller than the optimization time step - which is almost always the case - , several (many) iterations of the flow solver are performed between two optimization steps.

2.1.6 Moving mesh time steps

A good restriction to be imposed on the mesh movement in order to limit the apparition of flat or inverted elements is that vertices cannot cross too many elements on a single move between two mesh optimizations. Therefore, a geometric parameter CFL^{geom} is introduced to control the number of stages used to perform the mesh displacement between t and $t+\Delta t$. If CFL^{geom} is greater than one, the mesh is authorized to cross more than one element in a single move. In practice, CFL^{geom} is usually set between 1 and 8. The moving geometric time step is given by:

$$\delta t^{opt} = CFL^{geom} \max_{P_i} \frac{h_i}{\mathbf{v}_i}, \tag{2.3}$$

where h_i is the smallest height of all the elements in the ball of vertex P_i and $\mathbf{v}_i$ the mesh velocity at vertex P_i. In practice, when coupled with a flow solver, the actual time step is the minimum between the flow solver time step and the geometric one.

2.2 Arbitrary Lagrangian Eulerian flow solver

This section discusses in details the implementation of the Arbitrary Lagrangian Eulerian (ALE) flow solver `Wolf` that has been coupled to the moving mesh process described in Algorithm 5. After presenting the compressible Euler equations in the ALE framework, we describe the second order spatial discretization. As regards the temporal discretization, we first discuss the geometric conservation law (GCL) and then we present the explicit and implicit time integration schemes. This thesis has focused on the deriving and implementing implicit schemes in the framework of Mavriplis and Yang [Mavriplis 2006]. They are compared to explicit schemes in the last section.

2.2.1 Euler equations in the ALE framework

We consider the 3D unsteady compressible Euler equations for a Newtonian fluid in their ALE formulation set in the space-time computational domain $\mathcal{Q} = \Omega(t) \times [0, T]$, where T is the (positive) maximal time. The ALE formulation allows the equations to take arbitrary motion of the (domain) mesh into account. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the ALE formulation of the equations is written, for any arbitrary closed volume $C(t)$ of boundary $\partial C(t)$ moved with mesh velocity $\boldsymbol{w}$:

$$\begin{aligned} & \frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{C(t)} W \,\mathrm{d}\mathbf{x}\right) + \int_{\partial C(t)} (\mathcal{F}(W) - W \otimes \boldsymbol{w}) \cdot \mathbf{n} \,\mathrm{d}\mathbf{s} = \int_{C(t)} F_{ext} \,\mathrm{d}\mathbf{x} \\ \Longleftrightarrow\ & \frac{\mathrm{d}}{\mathrm{d}t}\left(\int_{C(t)} W \,\mathrm{d}\mathbf{x}\right) + \int_{\partial C(t)} \left(F\left(W\right) - W(\boldsymbol{w} \cdot \mathbf{n})\right) \,\mathrm{d}\mathbf{s} = \int_{C(t)} F_{ext} \,\mathrm{d}\mathbf{x}\,, \end{aligned}$$

where

$$\begin{cases} W = (\rho, \rho\mathbf{u}, \rho E)^T \text{ is the conservative variables vector} \\ \mathcal{F}(W) = (\rho\mathbf{u},\ \rho u\mathbf{u} + pe_1,\ \rho v\mathbf{u} + pe_2,\ \rho w\mathbf{u} + pe_3,\ (\rho E + p)\mathbf{u}) \text{ is the flux tensor} \\ F\left(W\right) = \mathcal{F}(W) \cdot \mathbf{n} = (\rho\eta,\ \rho u\eta + pn_1,\ \rho v\eta + pn_2,\ \rho w\eta + pn_3,\ (\rho E + p)\eta)^T \\ F_{ext} = (0,\ \rho\,\mathbf{f}_{ext},\ \rho\mathbf{u} \cdot \mathbf{f}_{ext})^T \text{ is the contribution of the external forces,} \end{cases}$$

and we have noted ρ the density of the fluid, p the pressure, $\mathbf{u} = (u, v, w)$ its Eulerian velocity, $\eta = \mathbf{u} \cdot \mathbf{n}$, $q = \|\mathbf{u}\|$, ε the internal energy per unit mass, $E = 1/2\, q^2 + \varepsilon$ the total energy per unit mass, $\mathbf{e} = (e_1, e_2, e_3)$ the canonical basis, $\mathbf{f}_{ext}$ the resultant of the volumic external forces applied on the particle and $\mathbf{n} = (n_1, n_2, n_3)$ the outward normal to interface $\partial C(t)$ of $C(t)$.

2.2.2 Spatial discretization

The spatial discretization of the governing equations is based on a vertex-centered finite volume formulation on unstructured meshes. This formulation consists in associating with each vertex of the mesh a control volume (or Finite-Volume cell) built by the rule of medians. `Wolf` proposes several choices of approximate Riemann solvers to compute numerical fluxes. Second-order space accuracy is achieved through a piecewise-linear extrapolation based on the Monotonic Upwind Scheme for Conservation Law (MUSCL) procedure with a particular edge-based formulation with upwind elements. Specific slope limiters are employed to damp or eliminate spurious oscillations that may occur in the vicinity of discontinuities. The main difference when translating these schemes from the standard formulation to the ALE formulation is the addition of the mesh velocities in the wave speeds of the Riemann solver problem.

Dual mesh construction

The spatial domain $\Omega(t)$ is discretized by a tetrahedral unstructured mesh $\mathcal{H}$. The vertex-centered finite volume formulation consists in associating a control volume denoted $C_i(t)$ with each vertex P_i of the mesh and at each time t. The dual finite volume cell mesh is built by the rule of medians. Discretized domain $\Omega_h(t)$ can be written as the union of the elements or the union of the finite volume cells:

$$\Omega_h(t) = \bigcup_{i=1}^{N_K} K_i(t) = \bigcup_{i=1}^{N_V} C_i(t)\,,$$

where N_K is the number of elements and N_V the number of vertices. Note that the dual mesh (composed of cells) is built in a preprocessing step. Consequently, only a simplicial mesh is needed in the input.

In 2D, this standard method consists in building cells bounded by segments of medians. Each triangle is split into three quadrangles (one associated with each one of its three vertices). The quadrangle's four vertices of the triangle associated with vertex P_i are:

- M_i, M_j: the middle of the two edges incident to P_i,
- G: the gravity center of the triangle,
- P_i: the considered vertex.

Median cell $C_i(t)$ of vertex P_i is the union of all quadrangles of the triangles surrounding P_i. An exemple of median cell is represented in Figure 2.2 (left). Consequently, at each edge P_iP_j of the mesh is associated a cell interface $\partial C_{ij}(t)$ which is common boundary between the two neighboring cells $C_i(t)$ and $C_j(t)$, *i.e.*, $\partial C_{ij}(t) = \partial C_i(t) \cap \partial C_j(t)$. Note that cell interface $\partial C_{ij}(t)$ is composed of two bi-segments (or one bi-segment for a boundary edge).

In 3D each tetrahedra is subdivided into four hexahedra (one associated with each one of its four vertices). The eight vertices of the hexahedron associated with a point P_i are given by:

- M_i, M_j, M_k: the middle of the three edges incident to P_i
- Gf_i, Gf_j, Gf_k: the gravity centers of the three faces containing P_i
- G: the gravity center of the tetrahedra
- P_i: the considered vertex.

Such an hexahedron is represented in Figure 2.2 (right). Median cell $C_i(t)$ of vertex P_i is the union of all hexahedra of the tetrahedra surrounding P_i. Consequently, at each edge P_iP_j of

the mesh is associated a cell interface $\partial C_{ij}(t)$ which is common boundary between the two neighboring cells $C_i(t)$ and $C_j(t)$, *i.e.*, $\partial C_{ij}(t) = \partial C_i(t) \cap \partial C_j(t)$. In three dimensions this cell interface $\partial C_{ij}(t)$ is composed of many triangular facets (two triangular facts per tetrahedra in the shell of edge P_iP_j).

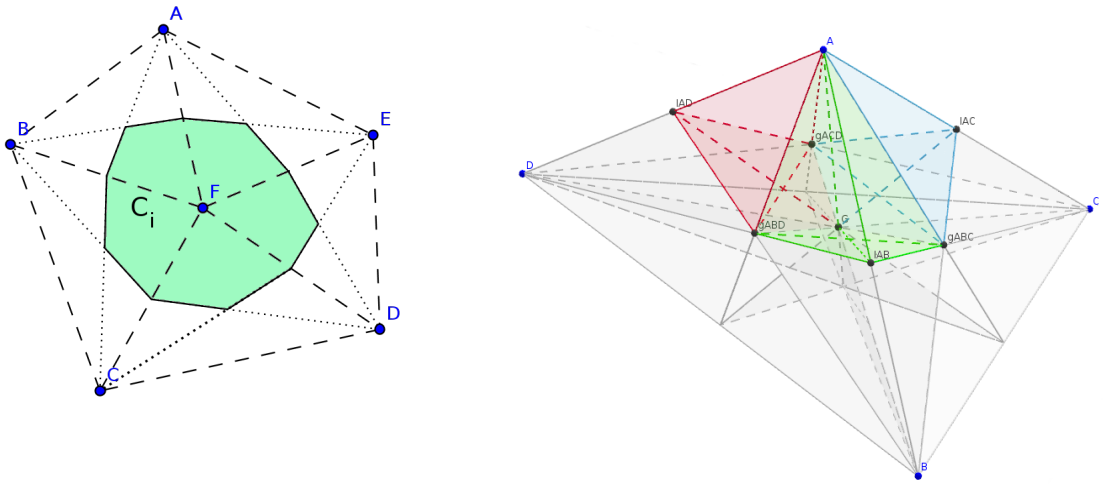


Figure 2.2: *Left, 2D finite volume median cells and interfaces made up of facets and, right, hexahedron associated with vertex P_i considered to build the finite volume cell of P_i.*

Edge-based Finite-Volume discretization

Assuming there is no contribution of the external forces, based on a finite volume formulation, the compressible Euler equations are integrated on each finite volume cell $C_i(t)$ (using the Green formula):

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(|C_i(t)|W_i(t)\right) + \mathbf{F}(W_i(t)) = 0\,, \tag{2.4}$$

where $W_i(t)$ is the mean value of the solution W on cell $C_i(t)$ and $F_i(W(t))$ is the numerical convective flux term:

$$\mathbf{F}(W_i(t)) = \int_{\partial C_i(t)} \left(\mathcal{F}(W_i(t)) - W_i(t) \otimes \boldsymbol{w}_i\right) \cdot \mathbf{n}_i \,\mathrm{d}\mathbf{s}\,, \tag{2.5}$$

where $\mathbf{n}_i$ is the outer normal to the finite volume cell surface $\partial C_i(t)$ as depicted in Figure 2.3, $\mathcal{F}$ is the convective term flux functions, and $\boldsymbol{w}_i$ is the mesh velocity at vertex P_i. The integration of convective fluxes $\mathbf{F}$ of Equation (2.5) is done by decomposing the cell boundary into many interfaces ∂C_{ij} composed of facets:

$$\mathbf{F}(W_i(t)) = \sum_{P_j \in \mathcal{V}(P_i)} \left(\mathcal{F}_{ij}(t) \cdot \int_{\partial C_{ij}(t)} \mathbf{n}_i \,\mathrm{d}\mathbf{s} - W_{ij}(t) \int_{\partial C_{ij}(t)} (\boldsymbol{w}_i \cdot \mathbf{n}_i)\,\mathrm{d}\mathbf{s} \right)\,,$$

where $\mathcal{V}(P_i)$ is the set of all neighboring vertices linked by an edge to P_i, and $W_{ij}(t)$ and $\mathcal{F}_{ij}(t)$ represents the constant value of W and $\mathcal{F}(W)$ at interface $\partial C_{ij}(t)$. The flow is calculated using a numerical flux function to approximate the flux at cell interface $\partial C_{ij}(t)$, denoted by Φ_{ij}:

$$\Phi_{ij}(t) = \Phi_{ij}(W_i(t), W_j(t), \mathbf{n}_{ij}(t), \sigma_{ij}(t)) \;=\; \mathcal{F}_{ij}(t)\cdot\mathbf{n}_{ij}(t) - W_{ij}(t)\sigma_{ij}(t)\,,$$

where

- $\mathbf{n}_{ij}(t) = \dfrac{1}{|\partial C_{ij}(t)|}\displaystyle\int_{\partial C_{ij}(t)} \mathbf{n}_i \,\mathrm{d}\gamma$ is the outward normalized normal (with respect to cell C_i) of cell interface ∂C_{ij},
- $\sigma_{ij}(t) = \dfrac{1}{|\partial C_{ij}(t)|}\displaystyle\int_{\partial C_{ij}(t)} \boldsymbol{w}_{ij}(t)\cdot\mathbf{n}_{ij}(t)\mathrm{d}\boldsymbol{s}$ is the normal velocity of cell interface $\partial C_{ij}(t)$.

We finally get the following semi-discretization at P_i:

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(|C_i(t)|W_i(t)\right) + \sum_{P_j\in\mathcal{V}(P_i)} |\partial C_{ij}(t)| \;\; \Phi_{ij}(W_i(t),\, W_j(t),\, \mathbf{n}_{ij}(t),\, \sigma_{ij}(t)) = 0\,, \tag{2.6}$$

The numerical flux function approximates the hyperbolic terms on the common boundary $\partial C_{ij}(t)$. We notice that the computation of the convective fluxes is performed monodimensionally in the direction normal to the boundary of the finite volume cell. Therefore, the numerical calculation of the flux function $\Phi_{ij}(t)$ at the interface $\partial C_{ij}(t)$ can be achieved by solving, at each time step, a one-dimensional Riemann problem in the direction of the normal $\mathbf{n}_{ij}$ by means of an approximate Riemann solver.

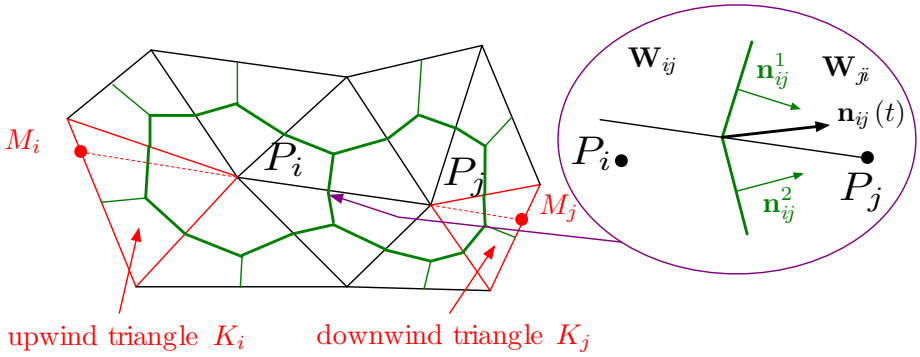


Figure 2.3: *Illustration of finite volume cells configuration around an edge P_iP_j in 2D: two neighboring cells C_i and C_j, their common interface ∂C_{ij} and the upwind triangles K_i and K_j associated with edge P_iP_j.*

Three approximate Riemann solvers are available in `Wolf` to compute the numerical flux Φ: Rusanov, Roe and HLLC. In the following, we describe their extensions to the ALE framework. For sake of clarity, the dependency in t of the variable is omitted.

Numerical flux computation

ALE Rusanov approximate Riemann solver. In the Rusanov's approach [Rusanov 1961], the upwinding term is defined thanks to the maximal spectral radius of the Jacobian matrix $\frac{\partial(\mathcal{F}(W)\cdot\mathbf{n})}{\partial W}$ given by: $\lambda = |\mathbf{u}\cdot\mathbf{n}| + c$. The flux resolution uses the following formulation of Φ:

$$\Phi^{Rusanov}(W_i, W_j, \mathbf{n}_{ij}) = \frac{\mathcal{F}(W_i)+\mathcal{F}(W_j)}{2}\cdot\mathbf{n}_{ij} + |\lambda_{ij}|\frac{W_i - W_j}{2}\,,$$

where $|\lambda_{ij}| = \max(|\mathbf{u}_i.\mathbf{n}_{ij}| + c_i\,,\, |\mathbf{u}_j.\mathbf{n}_{ij}| + c_j)$ and $c = \sqrt{\frac{\gamma p}{\rho}}$ is the speed of sound. For ALE simulation, the flux formulation takes into account the mesh displacement and is given by:

$$\begin{aligned}\Phi^{Rusanov}_{ale}(W_i, W_j, \mathbf{n}_{ij}, \sigma_{ij}) &= \frac{\mathcal{F}(W_i)\cdot\mathbf{n}_{ij} - \sigma_{ij}W_i + \mathcal{F}(W_j)\cdot\mathbf{n}_{ij} - \sigma_{ij}W_j}{2} + |\widetilde{\lambda}_{ij}|\frac{W_i - W_j}{2}\\ &= \frac{F_i + F_j}{2} - \sigma_{ij}\frac{W_i + W_j}{2} + |\widetilde{\lambda}_{ij}|\frac{W_i - W_j}{2}\end{aligned}$$

where $|\widetilde{\lambda}_{ij}| = \max(|\mathbf{u}_i.\mathbf{n}_{ij}| + c_i - \sigma_{ij}\,,\, |\mathbf{u}_j.\mathbf{n}_{ij}| + c_j - \sigma_{ij})$ and the notation $F_k = \mathcal{F}(W_k)\cdot\mathbf{n}_{ij}$.

ALE Roe approximate Riemann solver. In the Roe's approach [Roe 1981], the upwinding term is defined by the Jacobian matrix $A(W) = \frac{\partial(\mathcal{F}(W)\cdot\mathbf{n})}{\partial W}$. The eigenvalues of $A(W)$ are real and given by $|\mathbf{u}\cdot\mathbf{n}|$, $|\mathbf{u}\cdot\mathbf{n}| + c$ and $|\mathbf{u}\cdot\mathbf{n}| - c$, thus $A(W)$ is diagonalisable. In the context of the Euler equations, the hyperbolic flux is homogeneous of order one leading to the property:

$$\mathcal{F}(W)\cdot\mathbf{n} = A(W)\,W\,,$$

that allows non-oscillatory conservative schemes to be built. The flux resolution uses the formulation of Φ introduced by Roe:

$$\Phi^{Roe}(W_i, W_j, \mathbf{n}_{ij}) = \frac{\mathcal{F}(W_i)+\mathcal{F}(W_j)}{2}\cdot\mathbf{n}_{ij} + |\tilde{A}(W_i, W_j)|\frac{W_i - W_j}{2}\,,$$

where $\tilde{A}$ is the Jacobian matrix evaluated for the Roe's average variables and for all diagonalisable matrix $A = P\Lambda P^{-1}$ we have denoted $|A| = P|\Lambda|P^{-1}$. Let W_i and W_j be the two states, then the Roe's average variables are given by:

$$\begin{aligned}\tilde{\rho} &= \sqrt{\rho_i\rho_j}\\ \tilde{u} &= \frac{\sqrt{\rho_l}u_i + \sqrt{\rho_r}u_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}\\ \tilde{v} &= \frac{\sqrt{\rho_l}v_i + \sqrt{\rho_r}v_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}\\ \tilde{w} &= \frac{\sqrt{\rho_l}w_i + \sqrt{\rho_r}w_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}\\ \tilde{h} &= \frac{\sqrt{\rho_l}h_i + \sqrt{\rho_r}h_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}\end{aligned}$$

where h is the enthalpy per mass unit: $h = \frac{\rho E + p}{\rho}$, and from which we get

$$\tilde{c}^2 = (\gamma - 1)\left(\tilde{h} - \frac{1}{2}\tilde{q}^2\right) \quad \text{where} \quad \tilde{q}^2 = \tilde{u}^2 + \tilde{v}^2 + \tilde{w}^2$$

For ALE simulation, the flux formulation takes into account the mesh displacement:

$$\begin{aligned}\Phi^{Roe}_{ale}(W_i, W_j, \mathbf{n}_{ij}, \sigma_{ij}) &= \frac{\mathcal{F}(W_i)\cdot\mathbf{n}_{ij} - \sigma_{ij}W_i + \mathcal{F}(W_j)\cdot\mathbf{n}_{ij} - \sigma_{ij}W_j}{2} + |\tilde{A}(W_i, W_j) - \sigma_{ij}\mathcal{I}|\frac{W_i - W_j}{2} \\ &= \frac{F_i + F_j}{2} - \sigma_{ij}\frac{W_i + W_j}{2} + |\tilde{A}(W_i, W_j) - \sigma_{ij}\mathcal{I}|\frac{W_i - W_j}{2}\end{aligned}$$

where $\mathcal{I}$ is the identity matrix and with the notation $F_k = \mathcal{F}(W_k)\cdot\mathbf{n}_{ij}$.

HLLC numerical flux. The idea of the HLLC flow solver is to consider locally a simplified Riemann problem with two intermediate states depending on the local left and right states [Batten 1997]. The simplified solution to the Riemann problem consists of a contact wave with a velocity S_M and two acoustic waves, which may be either shocks or expansion fans. The acoustic waves have the smallest and the largest velocities (S_i and S_j, respectively) of all the waves present in the exact solution. If $S_i > 0$ then the flow is supersonic from left to right and the upwind flux is simply defined from $F(W_i)$ where W_i is the state to the left of the discontinuity. Similarly, if $S_j < 0$ then the flow is supersonic from right to left and the flux is defined from $F(W_j)$ where W_j is the state to the right of the discontinuity. In the more difficult subsonic case when $S_i < 0 < S_j$ we have to calculate $F(W_i^\star)$ or $F(W_j^\star)$. Consequently, the HLLC flux is given by:

$$\Phi^{HLLC}(W_i, W_j, \mathbf{n}_{ij}) = \begin{cases} \mathcal{F}(W_i)\cdot\mathbf{n}_{ij} & \text{if } S_i > 0 \\ \mathcal{F}(W_i^\star)\cdot\mathbf{n}_{ij} & \text{if } S_i \le 0 < S_M \\ \mathcal{F}(W_j^\star)\cdot\mathbf{n}_{ij} & \text{if } S_M \le 0 \le S_j \\ \mathcal{F}(W_j)\cdot\mathbf{n}_{ij} & \text{if } S_j < 0 \end{cases}.$$

where $W_i^\star$ and $W_j^\star$ are evaluated as follows. We denote by $\eta_k = \mathbf{u}_k\cdot\mathbf{n}_{ij}$. Assuming that $\eta^\star = \eta_i^\star = \eta_j^\star = S_M$, the following evaluations are proposed [Batten 1997]:

$$W_i^\star = \begin{cases} \rho_i^\star = \rho_i\dfrac{S_i - \eta_i}{S_i - S_M} \\ p_i^\star = p^\star = \rho_i(\eta_i - S_i)(\eta_i - S_M) + p_i \\ (\rho\mathbf{u})_i^\star = \dfrac{(S_i - \eta_i)\rho\mathbf{u}_i + (p^\star - p_i)\mathbf{n}_{ij}}{S_i - S_M} \\ (\rho E)_i^\star = \dfrac{(S_i - \eta_i)\rho E_i - p_i\eta_i + p^\star S_M}{S_i - S_M} \end{cases}, \quad W_j^\star = \begin{cases} \rho_j^\star = \rho_j\dfrac{S_j - \eta_j}{S_j - S_M} \\ p_j^\star = p^\star = \rho_j(\eta_j - S_j)(\eta_j - S_M) + p_j \\ (\rho\mathbf{u})_j^\star = \dfrac{(S_j - \eta_j)\rho\mathbf{u}_j + (p^\star - p_j)\mathbf{n}_{ij}}{S_j - S_M} \\ (\rho E)_j^\star = \dfrac{(S_j - \eta_j)\rho E_j - p_j\eta_j + p^\star S_M}{S_j - S_M} \end{cases},$$

A key feature of this solver is in the definition of the three waves velocity. For the contact wave we consider:

$$S_M = \frac{\rho_j \eta_j (S_j - \eta_j) - \rho_i \eta_i (S_i - \eta_i) + p_i - p_j}{\rho_j (S_j - \eta_j) - \rho_i (S_i - \eta_i)} ,$$

and the acoustic wave speeds based on the Roe average variable as defined above:

$$S_i = \min(\eta_i - c_i, \tilde{\eta} - \tilde{c}) \quad \text{and} \quad S_j = \max(\eta_j + c_j, \tilde{\eta} + \tilde{c})$$

where $\tilde{.}$ are Roe average variables [Roe 1981]. With such waves velocities, the approximate HLLC Riemann solver has the following properties. It automatically (i) satisfies the entropy inequality, (ii) resolves isolated contacts exactly, (iii) resolves isolated shocks exactly, and (iv) preserves positivity.

The methodology provided by Batten [Batten 1997] can be extended to the Euler equations in their ALE formulation:

$$\Phi_{ale}^{HLLC}(W_i, W_j, \mathbf{n}_{ij}, \sigma_{ij}) = \begin{cases} F(W_i) \cdot \mathbf{n}_{ij} - \sigma_{ij} W_i & \text{if } S_i > \sigma_{ij} \\ F(W_i^\star) \cdot \mathbf{n}_{ij} - \sigma_{ij} W_i^\star & \text{if } S_i \leq \sigma_{ij} < S_M \\ F(W_j^\star) \cdot \mathbf{n}_{ij} - \sigma_{ij} W_j^\star & \text{if } S_M \leq \sigma_{ij} \leq S_j \\ F(W_j) \cdot \mathbf{n}_{ij} - \sigma_{ij} W_j & \text{if } S_j < \sigma_{ij} \end{cases}$$

It is important to note that the considered waves speed (S_i, S_j and S_M) and the star states are the classical one and that the waves speed are compared in the moving frame.

Second-order accurate version

The previous formulation reaches at best a first-order spatial accuracy. A MUSCL type reconstruction method has been designed to increase the order of accuracy of the scheme. This method was introduced by Van Leer in a series of papers, see for instance [Leer 1972]. The idea is to use extrapolated values W_{ij} and W_{ji} instead of W_i and W_j at the interface ∂C_{ij} to evaluate the flux:

$$\Phi_{ij} = \Phi_{ij}(W_{ij}, W_{ji}, \mathbf{n}_{ij}, \sigma_{ij}) ,$$

and to achieve second-order accuracy.

In the implementation, we do not extrapolate directly the conservative variable W. In fact, the primitive variables $U = (\rho, u, v, w, p)$ are extrapolated to guarantee the positivity of the density and the pressure, then the conservative variables are reconstructed from these values. The extrapolation steps are the following:

In the following, we present how the MUSCL gradients are computed to design second-order accurate scheme with a fourth-order numerical dissipation. In contrast to the original

Algorithm 6 MUSCL extrapolation

1. Get primitive variables U_i and U_j from conservatives ones W_i and W_j
2. Compute primitive variable MUSCL gradients ∇U_{ij} and ∇U_{ji}
3. Extrapolates the primitive variables:

$$U_{ij} \;=\; U_i + \frac{1}{2}\,(\nabla U)_{ij}\cdot\overrightarrow{P_iP_j} \qquad \text{and} \qquad U_{ji} \;=\; U_j + \frac{1}{2}\,(\nabla U)_{ji}\cdot\overrightarrow{P_jP_i}\,.$$

4. Get conservative variables W_{ij} and W_{ji} from primitive ones U_{ij} and U_{ji}

MUSCL approach, the approximate "slopes" $(\nabla U)_{ij}$ and $(\nabla U)_{ji}$ are defined for any edge and are obtained using a combination of centered, upwind and nodal gradients.

The centered gradient related to edge $\overrightarrow{P_iP_j}$, is defined implicitly along edge $\overrightarrow{P_iP_j}$ via relation:

$$(\nabla U)^C_{ij}\;\overrightarrow{P_iP_j} = U_j - U_i\,. \tag{2.7}$$

Upwind and downwind gradients, which are also related to edge $\overrightarrow{P_iP_j}$, are computed using the upstream and downstream triangles/tetrahedra associated with this edge. These triangles/tetrahedra are respectively denoted K_{ij} and K_{ji}. K_{ij} (resp. K_{ji}) is the unique triangle/tetrahedron of the ball of P_i (resp. P_j) whose opposite face is crossed by the straight line prolongating edge $\overrightarrow{P_iP_j}$, see Figures 2.3 (2D) and 2.4 (3D). Upwind and downwind gradients of edge $\overrightarrow{P_iP_j}$ are then defined as:

$$(\nabla U)^U_{ij} = (\nabla U)_{|K_{ij}} \quad \text{and} \quad (\nabla U)^D_{ij} = (\nabla U)_{|K_{ji}}\,,$$

where

$$\nabla U_{|K} = \sum_{P_i\in K} (\nabla\phi_{P_i}\otimes U_i)$$

is the P^1-Galerkin gradient on element K and ϕ_{P_i} is the usual basis function associated with P_i. Parametrized nodal gradients are built by introducing the β-scheme:

$$\begin{aligned}(\nabla U)_{ij}\,\overrightarrow{P_iP_j} &= (1-\beta)\,(\nabla U)^C_{ij}\;\overrightarrow{P_iP_j} + \beta\;(\nabla U)^U_{ij}\;\overrightarrow{P_iP_j},\\ (\nabla U)_{ji}\,\overrightarrow{P_iP_j} &= (1-\beta)\,(\nabla U)^C_{ij}\;\overrightarrow{P_iP_j} + \beta\;(\nabla U)^D_{ij}\;\overrightarrow{P_iP_j}\,,\end{aligned}$$

where $\beta\in[0,1]$ is a parameter controlling the amount of upwinding. For instance, the scheme is centered for $\beta=0$ and fully upwind for $\beta=1$.

The most accurate β-scheme is obtained for $\beta = 1/3$, also called the V4-scheme. This scheme is third-order for the two-dimensional linear advection problem on structured triangular

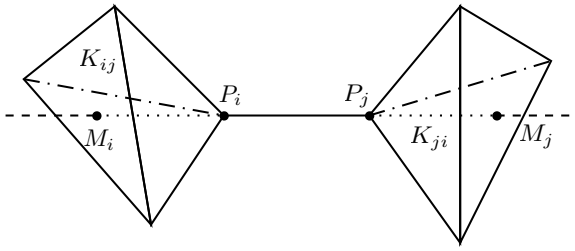


Figure 2.4: Downstream K_{ij} and upstream K_{ji} tetrahedra associated with edge $\overrightarrow{P_iP_j}$.

meshes. In our case, for the non-linear Euler equations on unstructured meshes, a second-order scheme with a fourth-order numerical dissipation is obtained [Debiez 2000]. The high-order gradients are given by:

$$(\nabla U)_{ij}^{V4}\ \overrightarrow{P_iP_j} = \tfrac{2}{3}\ (\nabla U)_{ij}^{C}\ \overrightarrow{P_iP_j} + \tfrac{1}{3}\ (\nabla U)_{ij}^{U}\ \overrightarrow{P_iP_j},$$
$$(\nabla U)_{ji}^{V4}\ \overrightarrow{P_iP_j} = \tfrac{2}{3}\ (\nabla U)_{ij}^{C}\ \overrightarrow{P_iP_j} + \tfrac{1}{3}\ (\nabla U)_{ij}^{D}\ \overrightarrow{P_iP_j}\,.$$

Limiter function

The aforementioned MUSCL scheme is not monotone and can be a source of spurious oscillations. These oscillations can affect the accuracy of the final solution or simply end the computation because (for instance) of negative pressures. A widely used technique for addressing this issue is to guarantee the TVD property in 1D [Harten 1983] or the LED property in 2D/3D of the scheme, which ensures that the extrapolated values W_{ij} and W_{ji} are not invalid. To guarantee the TVD or the LED properties, limiting functions are coupled with the previous high-order gradient evaluations. The gradient is substituted by a limited gradient denoted $(\nabla W)_{ij}^{lim}$. The choice of the limiting function is crucial as it directly affects the convergence of the simulation. Two limiters are possible with the V4-scheme.

Piperno Limiter. This limiter has been proposed by Piperno et al. [S. Piperno 1998]. Here, we present slightly modified version which is less dissipative. This limiter is expressed in a factorized form,

$$(\nabla U)_{ij}^{Lim}\ \overrightarrow{P_iP_j} = (\nabla U)_{ij}^{C}\ \overrightarrow{P_iP_j}\ \ \psi_{PI}\left(\frac{(\nabla U)_{ij}^{C}\ \overrightarrow{P_iP_j}}{(\nabla U)_{ij}^{V4}\ \overrightarrow{P_iP_j}}\right),$$

with

$$\psi_{PI}(R) = (\frac{1}{3} + \frac{2}{3}R)\begin{cases} \dfrac{3\frac{1}{R^2} - 6\frac{1}{R} + 19}{\frac{1}{R^3} - 3\frac{1}{R} + 18} & \text{if } R < 1 \\ 1 + (\dfrac{3}{2}\dfrac{1}{R} + 1)(\dfrac{1}{R} - 1)^3 & \text{if } R \geq 1 \end{cases}, \quad \text{where } R = \frac{(\nabla U)_{ij}^{C}\ \overrightarrow{P_iP_j}}{(\nabla U)_{ij}^{V4}\ \overrightarrow{P_iP_j}}.$$

Koren-Dervieux Limiter. The Koren-Dervieux limiter [Cournède 2006, Koren 1993] is a three-entry limiter which is a generalization of the SuperBee limiter for the V4-scheme. It reads:

$$(\nabla U)_{ij}^{lim} \overrightarrow{P_iP_j} = \psi_{KD}\left(\left(\nabla U^D\right)_{ij} \overrightarrow{P_iP_j},\ \left(\nabla U^C\right)_{ij} \overrightarrow{P_iP_j},\ \left(\nabla U^{V4}\right)_{ij} \overrightarrow{P_iP_j} \right) \tag{2.8}$$

with

$$\psi_{KD}(a,b,c) = \begin{cases} 0 & \text{if } ab \leq 0 \\ \text{sign}(a)\min(2|a|, 2|b|, |c|) & \text{otherwise} \end{cases}.$$

The operator ψ_{KD} defined above is applied component by component.

Boundary conditions

Boundary conditions are computed vertex-wise. Several conditions (inflow, outflow, symmetry, ...) are available in `Wolf`. We refer to [Alauzet 2010a] for their description. As, in this thesis, we are only concern with inviscid flows, the only boundary condition which is modified in the ALE framework is the slipping boundary condition applied to moving bodies. This boundary condition has to take into account the displacement of the body. To this end, we impose weakly [3]

$$\mathbf{u}_i \cdot \mathbf{n}_i = \sigma_i\,, \tag{2.9}$$

where $\mathbf{n}_i$ is the unitary boundary face normal and σ_i is the boundary face velocity. The standard ALE slipping boundary flux of vertex P_i reduces to:

$$\Phi_{ale}^{\text{slip}}(W_i, \mathbf{n}_i, \sigma_i) = \begin{pmatrix} 0 \\ -p_i \frac{\mathbf{n}_i}{\|\mathbf{n}_i\|} \\ -p_i \sigma_i \end{pmatrix}, \tag{2.10}$$

where p_i is the vertex pressure and $\mathbf{n}_i = \dfrac{\sum_{F_j \ni P_i} |F_j| \mathbf{n}_{F_j}}{\sum_{F_j \ni P_i} |F_j|}$ is the mean outward normal of the boundary interface. The F_j are the mesh boundary faces (edges/triangles). This flux is integrated over the boundary interface area $|\partial C_i|_\Gamma = \displaystyle\sum_{F_j \ni P_i} \frac{1}{3}|F_j|$, see Section 2.2.3.

However, when the second-order numerical schemes is considered, such a boundary condition creates oscillations in the density and the pressure when shock waves impact normally the boundary, see Figure 2.5. We thus prefer considering a mirror state and apply an approximate Riemann solver to diminish these oscillations. We thus have to evaluate the flux between the

[3] We do not enforce the numerical solution to verify $\mathbf{u}_i \cdot \mathbf{n}_i = \sigma_i$.

state on the boundary W and the ALE mirror state $\overline{W}$:

$$W_i = \begin{pmatrix} \rho_i \\ \rho \mathbf{u}_i \\ \rho E_i \end{pmatrix} \quad \text{and} \quad \overline{W}_i = \begin{pmatrix} \rho_i \\ \rho \mathbf{u}_i - 2\,\rho_i\,(\mathbf{u}_i \cdot \mathbf{n}_i - \sigma_i)\,\mathbf{n}_i \\ \rho E_i - 2\,\rho_i\,\sigma_i\,(\mathbf{u}_i \cdot \mathbf{n}_i - \sigma_i) \end{pmatrix}$$

as the mirror state verifies

$$\overline{p}_i = p_i, \quad \overline{c}_i = c_i, \quad \overline{\mathbf{u}}_i \cdot \overline{\mathbf{n}}_i = 2\,\sigma_i - \mathbf{u}_i \cdot \mathbf{n}_i \quad \text{and} \quad \overline{h}_i = h_i - 2\,\sigma_i\,(\mathbf{u}_i \cdot \mathbf{n}_i - \sigma_i)\,.$$

To evaluate the boundary flux, we consider the HLLC approximate Riemann solver between the state and the mirror state:

$$\Phi_{ale}^{\text{slip}}\,(W_i, \mathbf{n}_i, \sigma_i) = \Phi_{ale}^{\text{HLLC}}\left(W_i, \overline{W}_i, \mathbf{n}_i, \sigma_i\right)\,. \tag{2.11}$$

Note that by definition we have

$$\Phi_{ale}^{\text{HLLC}}(W_i, W_i, \mathbf{n}_i, \sigma_i) = \mathcal{F}(W_i) - \sigma_i W_i\,.$$

Thus, if Condition (2.9) is satisfied, then $W_i = \overline{W}_i$ and the flux $\Phi_{ale}^{\text{slip}}\,(W_i, \mathbf{n}_i, \sigma_i)$ simplifies to the form in Relation (2.10). In general, this condition is not satisfied, so we use Relation (2.11).

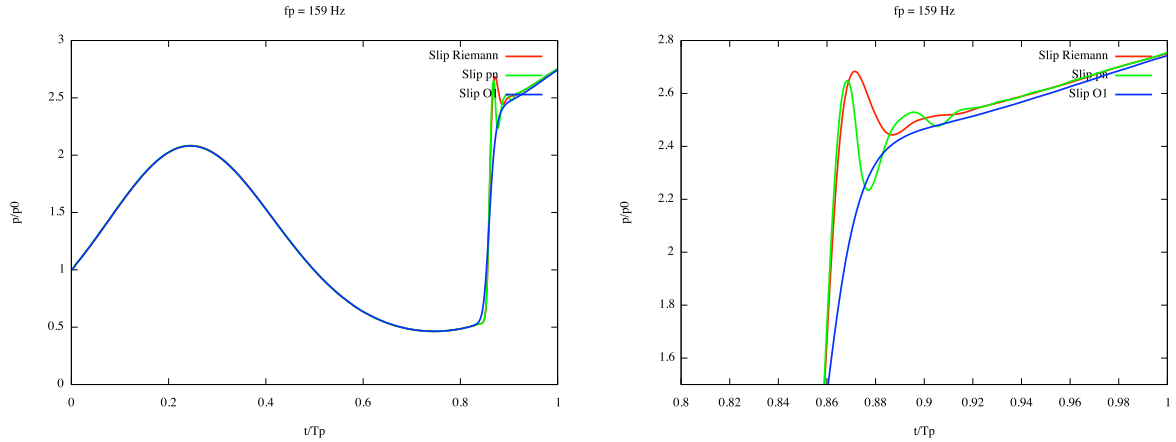


Figure 2.5: *Exemple of a piston problem where a shock wave impacts the pistion. The plot represents the pressure evolution in time at a sensor on the piston for the first-order scheme (blue curve) and the second-order scheme using the classical slip boundary condition (green curve) and the Riemann slip boundary condition (red curve).*

2.2.3 Time discretization

Temporal discretization is a more complex matter. In this section, we first recall the Geometric Conservation Law (GCL), and then, we described explicit time integration schemes. Finally, we presents the implicit time integration schemes that have been developed in this thesis. All the time discretizations are compliant with the Discrete Geometric Conservation Law (DGCL), which can be used to rigorously determine when the geometric parameters that appear in the fluxes should be computed.

The Geometric Conservation Law

We need to make sure that the movement of the mesh is not responsible for any artificial alteration of the physical phenomena involved, or at least, to make our best from a numerical point of view for the mesh movement to introduce an error of the same order as the one introduced by the numerical scheme. If System (2.4) is written for a constant state, assuming $F_{ext} = 0$, we get, for any arbitrary closed volume $C = C(t)$:

$$\frac{\mathrm{d}\left(|C(t)|\right)}{\mathrm{d}t} - \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n})\, \mathrm{d}\boldsymbol{s} = 0\,. \tag{2.12}$$

As the constant state is a solution of the Euler equations, if boundaries transmit the flux towards the outside as it comes, we find a purely geometrical relation inherent to the continuous problem. For any arbitrary closed volume $C = C(t)$ of boundary $\partial C(t)$, Relation (2.12) is integrated into:

$$|C(t+\delta t)| - |C(t)| = \int_t^{t+\delta t} \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n})\, \mathrm{d}\boldsymbol{s}\mathrm{d}t, \quad \text{with } t \text{ and } t+\delta t \in [0, T]\,, \tag{2.13}$$

which is usually known as the Geometric Conservation Law (GCL). From a geometric point of view, this relation states that the algebraic variation of the volume of C between two instants equals the algebraic volume swept by its boundary.

The role of the GCL in ALE simulations has been analyzed in [Etienne 2009]. It has been shown that the GCL is not a necessary condition to preserve time accuracy. However, violating it can lead to numerical oscillation [Mavriplis 2006]. In [Farhat 2001] the authors show that compliance with the GCL guarantees an accuracy of at least the first order in some conditions. Therefore, most would agree that the GCL should be enforced at the discrete level for a large majority of cases.

Discrete GCL enforcement

A new approach to enforcing the Discrete GCL was proposed in [Mavriplis 2006, Yang 2005, Yang 2007], in which the authors proposed a framework to build ALE high order temporal schemes that reach approximately the design order of accuracy. The originality of this approach

consists in precisely defining which ALE parameters are true degrees of freedom and which are not. In contrast to other approaches [Koobus 1999, Lesoinne 1996, Nkonga 2000], they consider that the times and configurations at which the fluxes are evaluated do not constitute a new degree of freedom to be set thanks to the ALE scheme. To maintain the design accuracy of the fixed-mesh temporal integration, the moment at which the geometric parameters, such as the cells' interfaces' normals or the upwind/downwind tetrahedra must be computed, is entirely determined by the intermediate configurations involved in the chosen temporal scheme. The only degree of freedom to be set by enforcing the GCL at the discrete level is σ. Incidentally, it is implicitly stated that $\boldsymbol{w}$ is never involved alone but only hidden in the term $\sigma\|\mathbf{n}\|$ which represents the instantaneous algebraic volume swept.
In practical terms, the interfaces normal speeds are found by simply rewriting the scheme for a constant discrete solution, which leads to a small linear system that is easily invertible by hand. This procedure is detailed in the next sections for one explicit Runge-Kutta scheme and several implicit schemes. Any fixed-mesh time-integration scheme can be extended to the case of moving meshes thanks to this methodology, and the resulting temporal scheme is naturally DGCL. Even if this has not been proven theoretically, the expected temporal order of convergence has also been observed numerically for several schemes designed using this method [Yang 2005].

Explicit Runge-Kutta schemes

The system of partial differential equations - given by Relation (2.6) - can be written in semi-discrete form for a vertex P_i as:

$$\frac{d\left(|C_i(t)|W_i(t)\right)}{dt} = \mathbf{F}(W_i(t))\,,$$

where $F(W_i(t))$ is the residual, *i.e.,* the discretization of the convective operator using an approximate Riemann solver and of the boundary conditions, and $|C_i|$ is the area/volume of the dual finite volume cell associated with P_i. For a first-order *explicit time discretization*, the semi-discretized system reads:

$$\frac{|C_i^{n+1}|W_i^{n+1} - |C_i^n|W_i^n}{\delta t_i^n} \;=\; \mathbf{F}(W_i^n)\,,$$

where W^n is the state at iteration n and δt_i^n is the local time step at iteration n. For high-order scheme in time Runge-Kutta scheme are considered.

Runge-Kutta (RK) methods are famous multi-stage methods to integrate ODEs. In the numerical solution of hyperbolic PDEs, notably the Euler equations, the favorite schemes among the huge family of Runge-Kutta schemes are those satisfying the Strong Stability Preserving

(SSP) property [Shu 1988, Spiteri 2002]. In what follows, we denote by SSPRK(S,P) the S-stage RK scheme of order P. We adopt the following notations:

$$\mathbf{F}(W_i^s) = \sum_{j=1}^{n_i} |\partial C_{ij}^s| \Phi_{ij}(W_i^s, W_j^s, \boldsymbol{n}_{ij}^s, \sigma_{ij}^s) = \sum_{j=1}^{n_i} \Phi_{ij}(W_i^s, W_j^s, \boldsymbol{\eta}_{ij}^s, \sigma_{ij}^s)$$

where

- n_i is the number of vertices P_j (edges $\mathbf{e}_{ij}$) in the ball of vertex P_i, *i.e.,* we have $P_j \in \mathcal{V}(P_i)$,
- $\boldsymbol{\eta}_{ij}$ is the outward **non-normalized** normal to the portion of the interface of cell C_i^s around edge $\mathbf{e}_{ij}$, *i.e.,* ∂C_{ij}, and we have: $\boldsymbol{\eta}_{ij} = |\partial C_{ij}| \mathbf{n}_{ij}$,
- σ_{ij} is the normal speed of the interface around edge $\mathbf{e}_{ij}$ of cell C_i^s.

Superscript notation X^s indicates that the quantity considered is the X obtained at stage s of the Runge-Kutta process. For instance, C_i^s is the cell associated with vertex P_i when the mesh has been moved to its s^{th} Runge-Kutta configuration.

In the following, coefficients $(c_s)_{0 \leq s \leq S}$ indicate the relative position in time of the current Runge-Kutta configuration: $t^s = t^n + c_s\, \delta t^n \quad$ with $\delta t^n = t^{n+1} - t^n$. Finally, we denote by A_{ij}^s the volume swept by the interface around edge $\mathbf{e}_{ij}$ of cell C_i between the initial Runge-Kutta configuration and the s^{th} configuration.

Application to the SSPRK(4,3) scheme. This approach was used, for example, to build the 3rd order 4-step Runge-Kutta scheme [Olivier 2011b], whose Butcher and Shu-Osher representations are given in Table 2.1. For this scheme to be DGCL, it must preserve a constant solution

Butcher representation	Shu-Osher representation	$t^s = t^n + c_s\, \delta t^n$
$Y_i^0 = Y_i^n$	$Y_i^0 = Y_i^n$	$c_0 = 0, \;\; t^0 = t^n$
$Y_i^1 = Y_i^0 + \frac{\delta t^n}{2} \mathbf{F}_i^0$	$Y_i^1 = Y_i^0 + \frac{\delta t^n}{2} \mathbf{F}_i^0$	$c_1 = \frac{1}{2}, \;\; t^1 = t^n + \frac{1}{2}\delta t^n$
$Y_i^2 = Y_i^0 + \frac{\delta t^n}{2} \left(\mathbf{F}_i^0 + \mathbf{F}_i^1\right)$	$Y_i^2 = Y_i^1 + \frac{\delta t^n}{2} \mathbf{F}_i^1$	$c_2 = 1, \;\; t^2 = t^{n+1}$
$Y_i^3 = Y_i^0 + \frac{\delta t^n}{6} \left(\mathbf{F}_i^0 + \mathbf{F}_i^1 + \mathbf{F}_i^2\right)$	$Y_i^3 = \frac{2}{3} Y_i^0 + \frac{1}{3} Y_i^2 + \frac{\delta t^n}{6} \mathbf{F}_i^2$	$c_3 = \frac{1}{2}, \;\; t^3 = t^n + \frac{1}{2}\delta t^n$
$Y_i^4 = Y_i^0 + \frac{\delta t^n}{2} \left(\frac{1}{3}\mathbf{F}_i^0 + \frac{1}{3}\mathbf{F}_i^1 + \frac{1}{3}\mathbf{F}_i^2 + \mathbf{F}_i^3\right)$	$Y_i^4 = Y_i^3 + \frac{\delta t^n}{2} \mathbf{F}_i^3$	$c_4 = 1, \;\; t^4 = t^{n+1}$

Table 2.1: *Butcher and Shu-Osher representations of the 3rd order 4-step Runge-Kutta scheme (SSPRK(4,3).*

$W_i = W_0$, as stated above. In this specific case, our conservative variable is $Y_i = |C_i| W_0$ and

the purely physical fluxes vanish, leading to $\mathbf{F}(W_i^s) = -W_0 \sum_{j=1}^{n_i} \boldsymbol{\eta}_{ij}^s \sigma_{ij}^s$. Therefore, the scheme reads:

$$
\begin{array}{rcl}
|C_i^0| & = & |C_i^n| \\
|C_i^1| - |C_i^0| & = & \displaystyle\sum_{j=1}^{n_i} A_{ij}^1 = \frac{\delta t^n}{2} \sum_{j=1}^{n_i} \boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 \\
|C_i^2| - |C_i^0| & = & \displaystyle\sum_{j=1}^{n_i} A_{ij}^2 = \frac{\delta t^n}{2} \sum_{j=1}^{n_i} \left(\boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 + \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1\right) \\
|C_i^3| - |C_i^0| & = & \displaystyle\sum_{j=1}^{n_i} A_{ij}^3 = \frac{\delta t^n}{6} \sum_{j=1}^{n_i} \left(\boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 + \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 + \boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2\right) \\
|C_i^4| - |C_i^0| & = & \displaystyle\sum_{j=1}^{n_i} A_{ij}^4 = \frac{\delta t^n}{2} \sum_{j=1}^{n_i} \left(\frac{1}{3}\boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 + \frac{1}{3}\boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 + \frac{1}{3}\boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2 + \boldsymbol{\eta}_{ij}^3 \sigma_{ij}^3\right),
\end{array}
$$

where A_{ij}^s is the volume swept by the interface around edge $\mathbf{e}_{ij}$ of cell C_i between the initial and the s^{th} Runge-Kutta configuration. A natural necessary condition for the above relations to be satisfied is to have, for each interface around edge $\mathbf{e}_{ij}$ of each finite volume cell C_i:

$$
\begin{pmatrix} A_{ij}^1 \\ A_{ij}^2 \\ A_{ij}^3 \\ A_{ij}^4 \end{pmatrix} = \delta t^n \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & 0 \\ \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{2} \end{pmatrix} \begin{pmatrix} \boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 \\ \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 \\ \boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2 \\ \boldsymbol{\eta}_{ij}^3 \sigma_{ij}^3 \end{pmatrix} \iff \begin{pmatrix} \boldsymbol{\eta}_{ij}^0 \sigma_{ij}^0 \\ \boldsymbol{\eta}_{ij}^1 \sigma_{ij}^1 \\ \boldsymbol{\eta}_{ij}^2 \sigma_{ij}^2 \\ \boldsymbol{\eta}_{ij}^3 \sigma_{ij}^3 \end{pmatrix} = \frac{1}{\delta t^n} \begin{pmatrix} 2 & 0 & 0 & 0 \\ -2 & 2 & 0 & 0 \\ 0 & -2 & 6 & 0 \\ 0 & 0 & -2 & 2 \end{pmatrix} \begin{pmatrix} A_{ij}^1 \\ A_{ij}^2 \\ A_{ij}^3 \\ A_{ij}^4 \end{pmatrix}.
$$

Therefore, the normal speed of the interface around edge $\mathbf{e}_{ij}$ of cell C_i must be updated in the Runge-Kutta process as follows :

$$
\sigma_{ij}^0 = \frac{2A_{ij}^1}{\delta t^n \ \boldsymbol{\eta}_{ij}^0}, \quad \sigma_{ij}^1 = \frac{-2A_{ij}^1 + 2A_{ij}^2}{\delta t^n \ \boldsymbol{\eta}_{ij}^1}, \quad \sigma_{ij}^2 = \frac{-2A_{ij}^2 + 6A_{ij}^3}{\delta t^n \ \boldsymbol{\eta}_{ij}^2}, \quad \text{and} \quad \sigma_{ij}^3 = \frac{-2A_{ij}^3 + 2A_{ij}^4}{\delta t^n \ \boldsymbol{\eta}_{ij}^3}, \tag{2.14}
$$

and the $\boldsymbol{\eta}_{ij}^s$ and A_{ij}^s are computed on the mesh once it has been moved to the s^{th} Runge-Kutta configuration.

Implicit time discretization

For numerical scheme stability, explicit time discretization have strong constraint on the allowable time-step defined by the CFL condition. Such explicit schemes are widely used for blast problems (violent flows), LES, aeroacoustic, ... to control efficiently dissipation and dispersion errors that can spoil the overall solution accuracy. However, for other applications, implicit scheme are of main interest to reduce the time-step restriction. But, implicit schemes are more difficult to implement as they require the solution of a linear system and their cost per iterations is generally very large w.r.t explicit schemes.

As above, the system of partial differential equations can be written in semi-discrete form for a vertex P_i as:

$$\frac{d\left(|C_i(t)|W_i(t)\right)}{dt} = \mathbf{F}(W_i(t))\,, \tag{2.15}$$

where $\mathbf{F}(W_i(t))$ is the residual and $|C_i(t)|$ is the area/volume of the dual finite volume cell associated with P_i at time t. To introduce the notations, we first assume we are not in the ALE framework and the finite volume cells have no dependency in time:

$$|C_i|\frac{d\left(W_i(t)\right)}{dt} = \mathbf{F}(W_i(t))\,.$$

Then, for a first order implicit time discretization (BDF1, *i.e.,* backward differentiation formula), we have :

$$\frac{|C_i|}{\delta t_i^n}(|W_i^{n+1} - W_i^n) \;=\; \mathbf{F}(W_i^{n+1})$$

which is linearized as:

$$\left(\frac{|C_i|}{\delta t_i^n} I_d - \frac{\partial \mathbf{F}}{\partial W}(W_i^n)\right)\left(W_i^{n+1} - W_i^n\right) \;=\; \mathbf{F}(W_i^n)$$

where $\dfrac{\partial \mathbf{F}}{\partial W}(W_i^n)$ is the Jacobian contributing to the i^{th} line of the matrix. We then rewrite the linearized system in compact form for all vertices in the mesh as:

$$\mathbf{A}^n\,\delta\mathbf{W}^n = \mathbf{R}^n$$

$$\text{where} \quad \mathbf{A}^n = \frac{|C|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial W} \quad \text{and} \quad \delta\mathbf{W}^n = \mathbf{W}^{n+1} - \mathbf{W}^n\,.$$

In this section, we present three implicit integration schemes in the ALE context: BDF1, BDF2, Crank-Nicolson, and the Defect Correction methods (DeC) to increase the spatial accuracy. Again, we follow the Mavriplis and Yang approach [Mavriplis 2006] and use the same notations as for the explicit case:

- n_i is the number of vertices P_j (edges $\mathbf{e}_{ij}$) in the ball of vertex P_i, *i.e.,* we have $P_j \in \mathcal{V}(P_i)$,
- $\boldsymbol{\eta}_{ij}$ is the outward **non-normalized** normal to the portion of the interface of cell C_i^s around edge $\mathbf{e}_{ij}$, *i.e.,* ∂C_{ij}, and we have: $\boldsymbol{\eta}_{ij} = |\partial C_{ij}|\mathbf{n}_{ij}$ and $\|\boldsymbol{\eta}_{ij}\| = |\partial C_{ij}|$ is the area of the portion of the cell interface,
- $\mathbf{n}_{ij} = \frac{\boldsymbol{\eta}_{ij}}{\|\boldsymbol{\eta}_{ij}\|}$ the normalized cell interface normal
- $\boldsymbol{\omega}_{ij}$ the cell interface velocity vector
- $\sigma_{ij} = \boldsymbol{\omega}_{ij} \cdot \mathbf{n}_{ij}$ is the normal velocity of the cell interface around edge $\mathbf{e}_{ij}$ of cell C_i^s, thus $\|\boldsymbol{\eta}_{ij}\|\,\sigma_{ij} = \boldsymbol{\omega}_{ij} \cdot \boldsymbol{\eta}_{ij}$ is the interface normal integrated velocity.

For each implicit time integration scheme, we give the normal velocity for each cell interface (as its computation depends on the considered scheme) and the linear system to be solved.

ALE BDF1 scheme. The first-order backward differentiation formula (BDF1) scheme reads:

$$\frac{|C^{n+1}|\mathbf{W}^{n+1} - |C^n|\mathbf{W}^n}{\delta t^n} = \mathbf{R}(\mathbf{W}^{n+1}, t^{n+1})\,, \tag{2.16}$$

the mesh coordinates and velocities in the residual should only be evaluated at the new time step t^{n+1}. Thus, we have:

$$\mathbf{R}(\mathbf{W}^{n+1}, t^{n+1}) = \mathbf{R}(\mathbf{W}^{n+1}, \mathbf{x}^{n+1}, \|\boldsymbol{\eta}^{n+1}\|\sigma^{n+1})\,.$$

As the mesh coordinates and the cell-face data should be taken at t^{n+1}, the only remaining degree of freedom is the mesh velocity $\boldsymbol{\omega}^{n+1}$ (or σ^{n+1}) and it is necessary to devise an implicit scheme which obeys the GCL. In fact, verifying the GCL defines completely the mesh velocity.

Computing σ. To compute the cells interface normal velocity, the GCL states that a constant state is preserved, thus for the BDF1 scheme the GCL reads:

$$|C_i^{n+1}| - |C_i^n| = \delta t^n \sum_{j=1}^{n_i} \|\boldsymbol{\eta}_{ij}^{n+1}\|\sigma_{ij}^{n+1}\,,$$

The left-handside, which can be computed exactly (following [Nkonga 1993]),but it may also be written as a linear combination of the incremental changes in volume between the two time levels, which themselves may be decomposed into the elemental volumes swept by each control volume boundary face between the two time levels. The equality is then required to hold for each moving control volume boundary face ∂C_{ij}, leading to:

$$\sigma_{ij}^{n+1} = \frac{|\partial C_{ij}^{n+1}| - |\partial C_{ij}^n|}{\|\boldsymbol{\eta}_{ij}^{n+1}\|\,\delta t^n}\,. \tag{2.17}$$

BDF1 linear system. As regards the BDF1 linear system, we linearize the residual of System (2.16) with respect to W^n and let x and σ fixed :

$$\begin{aligned}
&|C^{n+1}|\mathbf{W}^{n+1} - |C^n|\mathbf{W}^n = \delta t^n \mathbf{R}(\mathbf{W}^{n+1}, \mathbf{x}^{n+1}, \sigma^{n+1})\,,\\
\iff\quad &|C^{n+1}|\delta\mathbf{W}^n + (|C^{n+1}| - |C^n|)\mathbf{W}^n = \delta t^n \mathbf{R}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1}) + \delta t^n \frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1})\,\delta\mathbf{W}^n\,,
\end{aligned}$$

to obtain the following BDF1 linear system

$$\left(\frac{|C^{n+1}|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1})\right)\delta\mathbf{W}^n = \mathbf{R}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1}) - \frac{|C^{n+1}| - |C^n|}{\delta t^n}\mathbf{W}^n\,. \tag{2.18}$$

We notice that the Jacobian (in fact the whole matrix) and the residual are evaluated on the mesh at t^{n+1} but we use the solution at t^n.

In the unsteady case, the implicit scheme given by Equation (2.18) is A-stable but it is only first order accurate in time and in space. To increase the time accuracy to second order, two approaches are possible: the use of second-order backward difference formulae (BDF2) or the Crank-Nicolson method. To increase the space accuracy to second order a defect-correction (DeC) method is considered.

ALE BDF2 scheme. The second-order backward differentiation formula (BDF2) scheme is L-stable and uses three-points to integrate in time:

$$\frac{\lambda^{n+1}|C^{n+1}|\mathbf{W}^{n+1}+\lambda^{n}|C^{n}|\mathbf{W}^{n}+\lambda^{n-1}|C^{n-1}|\mathbf{W}^{n-1}}{\delta t^{n}}=\mathbf{R}(\mathbf{W}^{n+1},t^{n+1})\,, \tag{2.19}$$

where

$$\lambda^{n+1}=\frac{3}{2}\quad,\quad\lambda^{n}=-2\quad,\quad\lambda^{n-1}=\frac{1}{2}\,,$$

if the time-step δt is constant. If the time-step changes at each iteration, then we have [Denner 2014, Koobus 1999]:

$$\lambda^{n+1}=\frac{1+2r^{n}}{1+r^{n}}\quad,\quad\lambda^{n}=-1-r^{n}\quad,\quad\lambda^{n-1}=\frac{(r^{n})^{2}}{1+r^{n}}\quad\text{with}\quad r^{n}=\frac{\delta t^{n}}{\delta t^{n-1}}\,.$$

The mesh coordinates and velocities in the residual should only be evaluated at the new time step t^{n+1}. Thus, we have:

$$\mathbf{R}(\mathbf{W}^{n+1},t^{n+1})=\mathbf{R}(\mathbf{W}^{n+1},\mathbf{x}^{n+1},\|\boldsymbol{\eta}^{n+1}\|\sigma^{n+1})\,.$$

As the mesh coordinates and the cell-face data should be taken at t^{n+1}, the only remaining degree of freedom is the mesh velocity $\boldsymbol{\omega}^{n+1}$ (or σ^{n+1}) and it is necessary to devise an implicit scheme which obeys the GCL. In fact, verifying the GCL defines completely the mesh velocity.

Computing σ. The GCL states that a constant state is preserved, thus for the BDF2 scheme the GCL reads:

$$\lambda^{n+1}|C^{n+1}|+\lambda^{n}|C^{n}|+\lambda^{n-1}|C^{n-1}|=\delta t^{n}\sum_{j=1}^{n_i}\|\boldsymbol{\eta}_{ij}^{n+1}\|\sigma_{ij}^{n+1}\,,$$

The left-handside, which can be computed exactly (following [Nkonga 1993]), may also be written as a linear combination of the incremental changes in volume between the various time levels, which themselves may be decomposed into the elemental volumes swept by each control volume boundary face between time levels. The equality is then required to hold for each moving control volume boundary face, leading to:

$$\begin{aligned}&\delta t^{n}\|\boldsymbol{\eta}_{ij}^{n+1}\|\sigma_{ij}^{n+1}=\gamma^{n}\left(|\partial C_{ij}^{n+1}|-|\partial C_{ij}^{n}|\right)+\gamma^{n-1}\left(|\partial C_{ij}^{n}|-|\partial C_{ij}^{n-1}|\right)\\ \Longrightarrow\quad&\lambda^{n+1}|C^{n+1}|+\lambda^{n}|C^{n}|+\lambda^{n-1}|C^{n-1}|=\sum_{j=1}^{n_i}\left(\gamma^{n}\left(|\partial C_{ij}^{n+1}|-|\partial C_{ij}^{n}|\right)+\gamma^{n-1}\left(|\partial C_{ij}^{n}|-|\partial C_{ij}^{n-1}|\right)\right)\,,\end{aligned}$$

leading to $\gamma^{n}=\lambda^{n+1}$ and $\gamma^{n-1}=-\lambda^{n-1}$. And, we verify that:

$$\lambda^{n}=\gamma^{n-1}-\gamma^{n}=-\lambda^{n+1}-\lambda^{n-1}=\frac{-1-2r^{n}-(r^{n})^{2}}{1+r^{n}}=-(1+r^{n})=\lambda^{n}\,.$$

Consequently, the face normal integrated velocity is defined by the relation:

$$\|\boldsymbol{\eta}_{f}^{n+1}\|\sigma_{ij}^{n+1}=\frac{1}{\delta t^{n}}\left(\lambda^{n+1}\left(|\partial C_{ij}^{n+1}|-|\partial C_{ij}^{n}|\right)-\lambda^{n-1}\left(|\partial C_{ij}^{n}|-|\partial C_{ij}^{n-1}|\right)\right)\,,$$

and thus the cell face normal velocity is

$$\sigma_{ij}^{n+1} = \frac{\lambda^{n+1}\left(|\partial C_{ij}^{n+1}| - |\partial C_{ij}^{n}|\right) - \lambda^{n-1}\left(|\partial C_{ij}^{n}| - |\partial C_{ij}^{n-1}|\right)}{\|\boldsymbol{\eta}_{ij}^{n+1}\|\,\delta t^n}\,. \tag{2.20}$$

BDF2 linear system. To simplify the notation, the residual is now written $\mathbf{R}(\mathbf{W}^{n+1}, \mathbf{x}^{n+1}, \sigma^{n+1})$ meaning that it applies to the state at t^{n+1}, uses the mesh at t^{n+1} to compute all geometric quantities and considers σ^{n+1} defined above. The BDF2 scheme

$$\lambda^{n+1}|C^{n+1}|\mathbf{W}^{n+1} + \lambda^{n}|C^{n}|\mathbf{W}^{n} + \lambda^{n-1}|C^{n-1}|\mathbf{W}^{n-1} = \delta t^n \mathbf{R}(\mathbf{W}^{n+1}, \mathbf{x}^{n+1}, \sigma^{n+1})$$

is linearized as follows:

$$\begin{aligned}\lambda^{n+1}|C^{n+1}|\delta\mathbf{W}^{n} + (\lambda^{n+1}|C^{n+1}| + \lambda^{n}|C^{n}|)\mathbf{W}^{n} + \lambda^{n-1}|C^{n-1}|\mathbf{W}^{n-1} &= \delta t^n\, \mathbf{R}(\mathbf{W}^{n}, \mathbf{x}^{n+1}, \sigma^{n+1}) \\ &+ \delta t^n\, \frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^{n}, \mathbf{x}^{n+1}, \sigma^{n+1})\,\delta\mathbf{W}^{n}\,,\end{aligned}$$

to obtain the BDF2 linear system:

$$\begin{aligned}\left(\lambda^{n+1}\frac{|C^{n+1}|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^{n}, \mathbf{x}^{n+1}, \sigma^{n+1})\right)\delta\mathbf{W}^{n} &= \mathbf{R}(\mathbf{W}^{n}, \mathbf{x}^{n+1}, \sigma^{n+1}) \\ &- \frac{1}{\delta t^n}\left((\lambda^{n+1}|C^{n+1}| + \lambda^{n}|C^{n}|)\mathbf{W}^{n} + \lambda^{n-1}|C^{n-1}|\mathbf{W}^{n-1}\right)\,.\end{aligned} \tag{2.21}$$

We notice that the Jacobian (in fact the whole matrix) and the residual are evaluated on the mesh at t^{n+1} but we use the solution at t^n.

ALE Crank-Nicolson method. The Crank-Nicolson method (CN2) consists in using a general trapezoidal scheme:

$$\frac{|C^{n+1}|\mathbf{W}^{n+1} - |C^{n}|\mathbf{W}^{n}}{\delta t^n} = \beta\mathbf{R}(\mathbf{W}^{n+1}, t^{n+1}) + (1-\beta)\mathbf{R}(\mathbf{W}^{n}, t^{n})\,, \tag{2.22}$$

with $\beta = \frac{1}{2}$ (for $\beta = 0$ we recover the explicit scheme and for $\beta = 1$ we recover the Euler implicit time integration). This implicit scheme is A-stable and second order accurate in time. The mesh coordinates and velocities in the residual should be evaluated at the current time step t^n and the new time step t^{n+1}.

Computing σ. The GCL states that a constant state is preserved, thus for the CN2 scheme the GCL reads:

$$|C^{n+1}| - |C^{n}| = \frac{\delta t^n}{2}\sum_{j=1}^{n_i}\left(\|\boldsymbol{\eta}_{ij}^{n+1}\|\sigma_{ij}^{n+1} + \|\boldsymbol{\eta}_{ij}^{n}\|\sigma_{ij}^{n}\right)$$

Unfortunately here, we have two unknowns σ_{ij}^n and σ_{ij}^{n+1} for one equation. We make the following choice to solve this issue (as a choice has to be made):

$$|C^{n+1}| - |C^{n}| = \delta t^n \sum_{j=1}^{n_i}\|\boldsymbol{\eta}_{ij}^{n}\|\sigma_{ij}^{n} \quad \text{and} \quad |C^{n+1}| - |C^{n}| = \delta t^n \sum_{j=1}^{n_i}\|\boldsymbol{\eta}_{ij}^{n+1}\|\sigma_{ij}^{n+1}$$

Note that σ_{ij}^n and σ_{ij}^{n+1} will not be identical because the cell interface normal varies between t^n and t^{n+1}. Following, the results of the BDF1 scheme, we get:

$$\sigma_{ij}^n = \frac{|\partial C_{ij}^{n+1}| - |\partial C_{ij}^n|}{\|\boldsymbol{\eta}_{ij}^n\|\,\delta t^n} \quad \text{and} \quad \sigma_{ij}^{n+1} = \frac{|\partial C_{ij}^{n+1}| - |\partial C_{ij}^n|}{\|\boldsymbol{\eta}_{ij}^{n+1}\|\,\delta t^n}\,. \tag{2.23}$$

Crank-Nicolson linear system. The CN2 scheme:

$$|C^{n+1}|\mathbf{W}^{n+1} - |C^n|\mathbf{W}^n = \frac{\delta t^n}{2}\mathbf{R}(\mathbf{W}^{n+1}, \mathbf{x}^{n+1}, \sigma^{n+1}) + \frac{\delta t^n}{2}\mathbf{R}(\mathbf{W}^n, \mathbf{x}^n, \sigma^n)\,,$$

is linearized as follows:

$$|C^{n+1}|\delta\mathbf{W}^n + (|C^{n+1}| - |C^n|)\mathbf{W}^n = \frac{\delta t^n}{2}\mathbf{R}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1}) + \frac{\delta t^n}{2}\frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1})\,\delta\mathbf{W}^n + \frac{\delta t^n}{2}\mathbf{R}(\mathbf{W}^n, \mathbf{x}^n, \sigma^n)\,,$$

to obtain our linear system

$$\left(\frac{|C^{n+1}|}{\delta t^n}\mathbf{I} - \frac{1}{2}\frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1})\right)\delta\mathbf{W}^n = \frac{1}{2}\mathbf{R}(\mathbf{W}^n, \mathbf{x}^{n+1}, \sigma^{n+1}) + \frac{1}{2}\mathbf{R}(\mathbf{W}^n, \mathbf{x}^n, \sigma^n) - \frac{|C^{n+1}| - |C^n|}{\delta t^n}\mathbf{W}^n\,. \tag{2.24}$$

We note that in comparison to the fixed mesh case, the CN2 scheme require two residual evaluations, one on the mesh at t^n and the other one on the mesh at t^{n+1}.

Defect correction method. The BDF2 and the CN2 time integration schemes provide second order accurate schemes in time, but only first order accurate scheme in space. To recover the second order space convergence of the numerical scheme, the linearized linear system is solved by an iterative method proposed in [Martin 1996] and called Defect Correction (DeC). In the case of fixed meshes, the solution $\mathbf{W}^{n+1}$ is obtained after $\alpha_{\max}$ DeC iterations:

Set $\mathbf{W}^{n+1,1} = \mathbf{W}^n$

For $\alpha = 1, \ldots, \alpha_{\max}$ solve

$$\left(\frac{|C|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial W}\right)(\mathbf{W}^{n+1,\alpha+1} - \mathbf{W}^{n+1,\alpha}) = \mathbf{R}^n - \left(\frac{|C|}{\delta t^n}\mathbf{I} - \frac{\partial \mathbf{R}^n}{\partial W}\right)(\mathbf{W}^{n+1,\alpha} - \mathbf{W}^n)$$
$$= \mathbf{R}^{n+1,\alpha} - \frac{|C|}{\delta t^n}\left(\mathbf{W}^{n+1,\alpha} - \mathbf{W}^n\right)$$

EndFor

Set $\mathbf{W}^{n+1} = \mathbf{W}^{n+1,\alpha_{\max}}$

To recover the second order in time and in space, it is sufficient to couple the above methods. Now, we consider the case with dynamic meshes and we write the coupling of the DeC scheme with the BDF2 and CN2 schemes.

Coupling the DeC method with BDF2 leads to solve the following sub-iteration linear system:

$$\left(\lambda^{n+1}\frac{|C^{n+1}|}{\delta t^n}\mathbf{I}-\frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^n,\mathbf{x}^{n+1},\sigma^{n+1})\right)\left(\mathbf{W}^{n+1,\alpha+1}-\mathbf{W}^{n+1,\alpha}\right)$$
$$= \mathbf{R}(W^{n+1,\alpha},\mathbf{x}^{n+1},\sigma^{n+1})-\lambda^{n+1}\frac{|C^{n+1}|}{\delta t^n}\left(\mathbf{W}^{n+1,\alpha}-\mathbf{W}^n\right)$$
$$-\frac{1}{\delta t^n}\left((\lambda^{n+1}|C^{n+1}|+\lambda^n|C^n|)\mathbf{W}^n+\lambda^{n-1}|C^{n-1}|\mathbf{W}^{n-1}\right).$$

Coupling the DeC method with Crank-Nicolson leads to solve the following sub-iteration linear system:

$$\left(\frac{|C^{n+1}|}{\delta t^n}\mathbf{I}-\frac{1}{2}\frac{\partial \mathbf{R}}{\partial W}(\mathbf{W}^n,\mathbf{x}^{n+1},\sigma^{n+1})\right)(\mathbf{W}^{n+1,\alpha+1}-\mathbf{W}^{n+1,\alpha})$$
$$= \frac{1}{2}\mathbf{R}(\mathbf{W}^n,\mathbf{x}^n,\sigma^n)+\frac{1}{2}\mathbf{R}(\mathbf{W}^n,\mathbf{x}^{n+1},\sigma^{n+1})-\frac{|C^{n+1}|-|C^n|}{\delta t^n}\mathbf{W}^n$$
$$-\left(\frac{|C^{n+1}|}{\delta t^n}\mathbf{I}-\frac{1}{2}\frac{\partial \mathbf{R}^n}{\partial W}(\mathbf{W}^n,\mathbf{x}^{n+1},\sigma^{n+1})\right)(\mathbf{W}^{n+1,\alpha}-\mathbf{W}^n)$$
$$= \frac{1}{2}\mathbf{R}(\mathbf{W}^n,\mathbf{x}^n,\sigma^n)+\frac{1}{2}\mathbf{R}(\mathbf{W}^{n+1,\alpha},\mathbf{x}^{n+1},\sigma^{n+1})-\frac{|C^{n+1}|-|C^n|}{\delta t^n}\mathbf{W}^n-\frac{|C^{n+1}|}{\delta t^n}(\mathbf{W}^{n+1,\alpha}-\mathbf{W}^n)$$
$$= \frac{1}{2}\mathbf{R}(\mathbf{W}^n,\mathbf{x}^n,\sigma^n)+\frac{1}{2}\mathbf{R}(\mathbf{W}^{n+1,\alpha},\mathbf{x}^{n+1},\sigma^{n+1})-\frac{1}{\delta t^n}\left(|C^{n+1}|\mathbf{W}^{n+1,\alpha}-|C^n|\mathbf{W}^n\right).$$

Practical implementation. In practice, the following steps are done in the implementation:

1. We compute the volume swept by the cell faces using Nkonga formula, see next Section (with the cell-face velocity $\boldsymbol{\omega}_{ij}$ and the cell-face non-normalized pseudo-normal $\tilde{\boldsymbol{\eta}}_{ij}$),
2. We compute the cell-face normal velocity σ_{ij} using Equation (2.17) or (2.20) or (2.23)
3. Numerical fluxes can be evaluated using σ_{ij}.

Practical computation of the volumes swept

The interface of a finite volume cell is made up of several triangles, connecting the middle of an edge to the center of gravity of a face and the center of gravity of that tetrahedron, see Figure 2.6. The two triangles of the interface sharing one edge within a tetrahedron are coplanar (*i.e.* the middle of an edge, the center of gravity of the two faces neighboring this edge and the center of gravity of tetrahedron are coplanar). The union of these two triangles is called the facet associated to the edge and the tetrahedron.

At configuration $t^s = t^n + c_s\,\delta t^n$, (with $t^{n+1} = t^n + \delta t^n$), the outward non-normalized pseudo normal $\tilde{\boldsymbol{\eta}}^s_{ij}$ and the volume swept A^s_{ij} are computed as described in [Nkonga 1994]. As the cell interface is made up of several facets, the total swept volume is the sum of the volumes swept

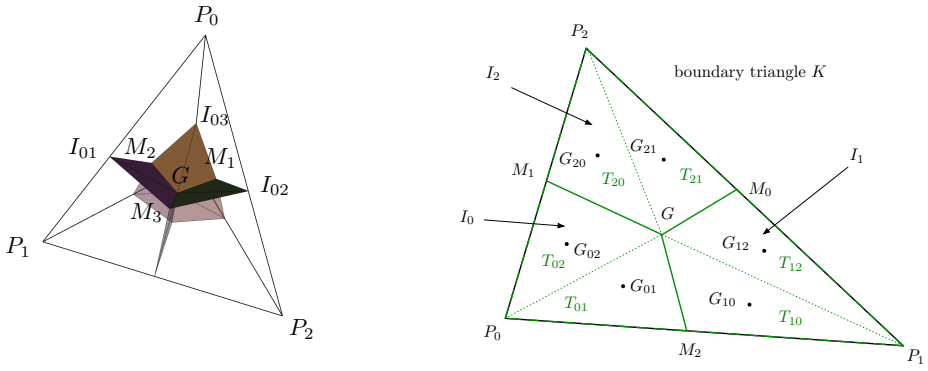


Figure 2.6: Interface of a finite volume cell made up of facets (top) and boundary interface (bottom).

by each facet. Let us assume that the facet considered is associated with edge $\mathbf{e}_{ij} = P_iP_j$ and belongs to tetrahedron $K = (P_0, P_1, P_2, P_3)$. In what follows, $i \neq j \neq l \neq m \in [\![0, 3]\!]$, G denotes the center of gravity of tetrahedron K, M_m denotes the gravity center of face $F_m = (P_i, P_j, P_l)$ of tetrahedron K and M_l the center of gravity of face $F_l = (P_i, P_j, P_m)$. The outward non-normalized pseudo normal of the facet is given by:

$$\tilde{\boldsymbol{\eta}}^s_{ij,K} = \frac{1}{4}G^nM^n_m \wedge G^sM^s_l + \frac{1}{4}G^sM^s_m \wedge G^nM^n_l + \frac{1}{2}G^nM^n_m \wedge G^nM^n_l + \frac{1}{2}G^sM^s_m \wedge G^sM^s_l \,, \tag{2.25}$$

where

$$G^nM^n_m = \frac{1}{12}\left(P^n_i + P^n_j - 3P^n_m + P^n_l\right) \,, \quad G^sM^s_m = \frac{1}{12}\left(P^s_i + P^s_j - 3P^s_m + P^s_l\right) \,,$$
$$G^nM^n_l = \frac{1}{12}\left(P^n_i + P^n_j + P^n_m - 3P^n_l\right) \,, \quad G^sM^s_l = \frac{1}{12}\left(P^s_i + P^s_j + P^s_m - 3P^s_l\right) \,.$$

The volume swept by the facet is:

$$A^s_{ij,K} \;=\; c_s\,\delta t^n\,(\boldsymbol{w}_G)^s_{ij,K}\cdot\tilde{\boldsymbol{\eta}}^s_{ij,K}\,, \tag{2.26}$$

with the mean velocity written as:

$$(\boldsymbol{w}_G)^s_{ij,K} = \frac{1}{36\,c_s\,\delta t^n}\left(13\boldsymbol{w}^s_i + 13\boldsymbol{w}^s_j + 5\boldsymbol{w}^s_m + 5\boldsymbol{w}^s_l\right)$$

with

$$\boldsymbol{w}^s_\xi = P^s_\xi - P^n_\xi \,.$$

Finally, the total volume swept by the interface around edge $\mathbf{e}_{ij}$ of cell C_i is obtained by summing over the shell of the edge, *i.e.* all the tetrahedra sharing the edge:

$$A^s_{ij} = \sum_{K\in Shell(i,j)} A^s_{ij,K} \,. \tag{2.27}$$

It is important to understand that normals $\tilde{\boldsymbol{\eta}}^s_{ij,K}$ are pseudo-normals which are used only to compute the volumes swept by the facets. They must not be mistaken for normals to facets taken at t^s, $\boldsymbol{\eta}^s_{ij,K}$, which are used for the computation of ALE fluxes.

Volumes swept by boundary interfaces

The pseudo-normals and swept volumes of boundary faces are computed in a similar way. Let $K = (P_0\,, P_1\,, P_2)$ be a boundary triangle, as in Figure 2.6. Let M_0, M_1 and M_2 be the middles of the edges and G the center of gravity of K. The triangle is made up of three quadrangle finite volume interfaces: $(P_0\,, M_2\,, G\,, M_1)$ associated with cell C_0, $(P_1\,, M_0\,, G\,, M_2)$ with C_1 and $(P_2\,, M_1\,, G\,, M_0)$ with C_2. Each boundary quadrangle interface I_i is made of two sub-triangles, noted T_{ij} and T_{ik} with $j, k \neq i$.

The volume swept by interface I_i between the initial and current configurations is the sum of the volumes swept by its two sub-triangles:

$$A^s_{i,K} = A^s_{T_{ij}} + A^s_{T_{ik}} \;=\; c_s \delta t^n\, \boldsymbol{w}^s_{G_{ij}} \cdot \tilde{\boldsymbol{\eta}}^s_{ij} \;+\; c_s \delta t^n\, \boldsymbol{w}^s_{G_{ik}} \cdot \tilde{\boldsymbol{\eta}}^s_{ik}\,, \tag{2.28}$$

where $\boldsymbol{w}^s_{G_{ij}}$ is the velocity of the center of gravity G_{ij} of triangle T_{ij} and $\tilde{\boldsymbol{\eta}}^s_{ij}$ is the pseudo-normal associated with T_{ij}, computed between the initial and current configurations. The six triangles T_{ij} are coplanar, so their pseudo-normals have the same direction. Moreover, as median cells are used, their pseudo-normals also have the same norm, which is equal to one sixth of the norm of the pseudo-normal to triangle K. This common pseudo-normal is therefore equal to:

$$\tilde{\boldsymbol{\eta}}^s \;=\; \frac{1}{6}\tilde{\boldsymbol{\eta}}^s_K \;=\; \frac{1}{6}\cdot\frac{1}{3}\left[\frac{1}{4}P^n_0P^n_1 \wedge P^s_0P^s_2 + \frac{1}{4}P^s_0P^s_1 \wedge P^n_0P^n_2 + \frac{1}{2}P^n_0P^n_1 \wedge P^n_0P^n_2 + \frac{1}{2}P^s_0P^s_1 \wedge P^s_0P^s_2\right], \tag{2.29}$$

thus

$$A^s_{i,K} \;=\; c_s \delta t^n \left[\boldsymbol{w}^s_{G_{ij}} + \boldsymbol{w}^s_{G_{ik}}\right] \cdot \tilde{\boldsymbol{\eta}}^s\,, \tag{2.30}$$

where :

$$\boldsymbol{w}^s_{G_{ij}} = \frac{1}{18\, c_s\, \delta t^n}\left(11\boldsymbol{w}^s_i + 5\boldsymbol{w}^s_j + 2\boldsymbol{w}^s_k\right) \quad \text{with} \quad \boldsymbol{w}^s_\xi = P^s_\xi - P^n_\xi\,.$$

Finally, the total volume swept by the boundary interface is:

$$A^s_i = \sum_{K \in Ball(i)} A^s_{i,K}\,. \tag{2.31}$$

MUSCL approach and temporal schemes

Regarding spatial accuracy, we have seen that the order of accuracy can be enhanced using the MUSCL-type reconstruction with upwinding. However, in the ALE context, one must determine how and when upwind/downwind elements should be evaluated to compute upwind/downwind

gradients which are necessary for the β-schemes. This question is neither answered in the literature and generally approximations are carried out. For instance, some papers propose to use upwind and downwind elements at t^n for the whole Runge-Kutta process. However, this choice should be consistent with the considered time integration scheme. Following the framework of [Yang 2005], it is clear that preserving the expected order of accuracy in time imposes that the upwind/downwind elements and the gradients are computed on the current configuration, *i.e.,* on the mesh at t^s for the Runge-Kutta stage or the mesh specified in the residual for implicit schemes. Therefore, similarly to geometric parameters, the upwind/downwind elements and the gradients should be re-evaluated at each step of the Runge-Kutta stage or at each time step for implicite schemes.

Computation of the time step

For explicit time integration : The maximal allowable time step for the numerical scheme is:

$$\delta t^n(P_i) = \frac{h(P_i)}{c_i + \|\mathbf{u}_i - \boldsymbol{w}_i\|} \tag{2.32}$$

where $h(P_i)$ is the smallest height in the ball of vertex P_i, c_i is the speed of sound, $\mathbf{u}_i$ is the Eulerian speed (the speed of the fluid, computed by the solver) and $\boldsymbol{w}_i$ is the speed of the mesh vertex. The global time step is then given by $\delta t^n = CFL\ \min_{P_i}(\delta t^n(P_i))$.

For implicit time integration : Time step is up to the user.

Handling the swaps

An ALE formulation of the swap operator, satisfying the DGCL, was proposed in 2D in [Olivier 2011b], but its extension to 3D is very delicate because it requires to handle 4D geometry, and it has not been carried out yet. Instead of considering an ALE scheme for the swap operator, our choice in this work is to perform the swaps between two solver iterations, *i.e.* at a fixed time t^n. This consequently means that during the swap phase, the mesh vertices do not move, and thus the swaps do not impact the ALE parameters η and $\boldsymbol{w}$, unlike [Olivier 2011b] where the swaps are performed during the solver iteration, *i.e.* between t^n and t^{n+1}. After each swap, the solution should be updated on the new configuration. Two interpolation methods are considered here.

The first, and simplest, one is to perform a linear interpolation to recover the solution. As only the connectivity changes and not the vertices positions, the solution at the vertices does not change, *i.e.,* nothing has to be done. This interpolation is DGCL compliant, since the constant state is preserved (in fact, any linear state is preserved), but it does not conserve the

mass (*i.e.,* it does not conserve the integral of the conservative variable) which is problematic for conservative equations when discontinuities are involved in the flow.

The second method is the P^1-exact conservative interpolation following [Alauzet 2010b, Alauzet 2016]. It is a simplified version of the latter because the cavity of the swap configuration is fixed. The mass conservation property of the interpolation operator is achieved by element-element intersections. The idea is to find, for each element of the new configuration, its geometric intersection with all the elements of the previous configuration it overlaps and to mesh this geometric intersection with simplices. We are then able to use a Gauss quadrature formula to exactly compute the mass which has been locally transferred. High-order accuracy is obtained through the reconstruction of the gradient of the solution from the discrete data and the use of some Taylor formulae. Unfortunately, this high-order interpolation can lead to a loss of monotonicity. The maximum principle is recovered by correcting the interpolated solution in a conservative manner, using a limiter strategy very similar to the one used for Finite-Volume solvers. Finally, the solution values at vertices are reconstructed from this piecewise linear by element discontinuous representation of the solution. The algorithm is summarized in Algorithm 9 where m_K stands for the integral of any conservative quantities (density, momentum and energy) on the considered element. This method is also compliant with the DGCL.

Algorithm 7 Conservative Interpolation Process

1. For all elements K_{back} of the original cavity, compute solution mass $m_{K_{back}}$ and gradient $\nabla_{K_{back}}$
2. For all elements K_{new} of the new cavity, recover solution mass $m_{K_{new}}$ and gradient $\nabla_{K_{new}}$:
   (a) compute the intersection of K_{new} with all K^i_{back} it overlaps
   (b) mesh the intersection polygon/polyhedra of each couple of elements (K_{new}, K^i_{back})
   (c) compute $m_{K_{new}}$ and $\nabla_{K_{new}}$ using Gauss quadrature formulae

   $\Longrightarrow$ a piecewise linear discontinuous representation of the mass on the new cavity is obtained
3. Correct the gradient to enforce the maximum principle
4. Set the solution values to vertices by an averaging procedure

Moreover, after each swap, the data of the finite volume cells (volume and interface normals) are updated, together with the topology of the mesh (edges and tetrahedra). This requires to have a flow solver with dynamic data.

2.3 FSI coupling

The moving boundaries can have an imposed motion, or be driven by fluid-structure interaction. A simple solid mechanics solver is coupled to the flow solver described previously. The chosen approach is the 6-DOF (6 Degrees of Freedom) approach for rigid bodies.

In this work, the bodies are assumed to be rigid, of constant mass and homogeneous, *i.e.,* their mass is uniformly distributed in their volume. The bodies we consider will never break into different parts. Each rigid body B is fully described by:

Physical quantities: its boundary ∂B and its associated inward normal $\mathbf{n}$, its mass m assumed to be constant, its $d \times d$ matrix of inertia $\mathcal{J}_G$ computed at G which is symmetric and depends only on the shape and physical nature of the solid object[4].

Kinematic quantities: the position of its center of gravity $\mathbf{x_G} = (x(t), y(t), z(t))$, its angular displacement vector $\boldsymbol{\theta} = \boldsymbol{\theta}(t)$ and its angular speed vector $\boldsymbol{\omega} = \mathrm{d}\boldsymbol{\theta}/\mathrm{d}t$.

$\mathbf{F}_{ext}$ denotes the resultant vector of the external forces applied on B, $\mathbf{M}_G(\mathbf{F}_{ext})$ the kinetic moment of the external forces applied on B computed at G and $\mathbf{g}$ the gravity vector. We assume that the bodies are only submitted to forces of gravity and fluid pressure. The equations for solid dynamics in an inertial frame then read:

$$\left\{ \begin{array}{rcll} m\dfrac{\mathrm{d}^2\mathbf{x_G}}{\mathrm{d}t^2} & = & \mathbf{F}_{ext} & = \displaystyle\int_{\partial B} p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\mathrm{d}\boldsymbol{s} + m\mathbf{g} \\ \mathcal{J}_G\dfrac{\mathrm{d}^2\boldsymbol{\theta}}{\mathrm{d}t^2} & = & \mathbf{M}_G(\mathbf{F}_{ext}) & = \displaystyle\int_{\partial B} \big[(\boldsymbol{s} - \mathbf{x_G}) \times p(\boldsymbol{s})\mathbf{n}(\boldsymbol{s})\big]\mathrm{d}\boldsymbol{s}\,. \end{array} \right. \tag{2.33}$$

The equation governing the position of the center of gravity of the body is easy to solve since it is linear. Its discretization is straightforward. However, the discretization of the second equation, which controls the orientation of the body, is more delicate. Since the matrix of inertia $\mathcal{J}_G$ depends on $\boldsymbol{\theta}$, it is a non-linear second order ODE. The chosen discretization is extensively detailed in [Olivier 2011a] and is based on rewriting of the equations in the frame of the moving body.

As the geometry must be moved in accordance with the fluid computation, the same time integration scheme has been taken to integrate the fluid and the solid equations. Therefore, time-advancing of the rigid bodies ODE System is performed using the same RKSSP or implicit scheme as the one used to advance the fluid numerical solution. The coupling is loose and explicit as the external forces and moments acting on rigid objects are computed on the current configuration.

[4]The moment of inertia relative to an axis of direction $\mathbf{a}$ passing through G where $\mathbf{a}$ is an arbitrary unit vector is given by: $J_{(G\mathbf{a})} = \mathbf{a}^T \mathcal{J}_G \mathbf{a}$

2.4 Numerical experiments

2.4.1 Naca0012 Pitching

The formation of vortices in the wake of an aircraft is essentially the result of the separation of flow and the vorticity created by the lifting surfaces. Air below the wing is at higher pressure than the air pressure above. This pressure difference causes air to flow from the lower surface of the wing, around the wingtip towards the upper surface of the wing. This phenomenon creates a vortex which is lift-generating. But wingtip vortices in lift generation are also associated to wake turbulence and increase the drag of the aircraft. This could be particularly dangerous during take-off, landing or when the aircraft operates a high-angle of attack. Vortical flows, if not properly understood and controlled, may be undesirable aerodynamic effects and can severely influence the flight stability or speed up the degradation of the aircraft structure. Consequently, there is a continuous need for better understanding of vortex flow characteristics to predict the flow behavior and to help the designers to develop more effective means.

Mesh

In a circular domain of radius 20 is put a NACA0012 of length 1. The entire mesh includes 3291 vertices.

Initial conditions

To generate similar conditions of take-off, the initial Mach number chosen is 0.4.

Moving mesh

The moving of the NACA0012 is imposed. In phase 1, the coordinates of the geometry do not moved. The flow around the aircraft is established. Let $t_1 = 0.1$ be the time of the beginning of the movement and $\theta^{max} = 20^o$ the maximal angle reached at first time $t_{\theta^{max}}$. The movement of the wing profile is defined in phase 2 by calculating at each time step its angle of rotation $\alpha = 0.5 * \theta^{max} * (1 - \cos(\pi * (t - t_1)/(t_{\theta^{max}} - t_0)))$. The coordinates of the new points are then $x_{new} = x_{old} * \cos\alpha + y_{old} * \sin\alpha$ and $y_{new} = -x_{old} * \sin\alpha + y_{old} * \cos\alpha$. A phase 3 follows where the NACA0012 stays in upward position before the descent computed in the same way as phase 2.

Numerical results and comparison between implicit and explicit solvers

The simulation is done both in explicit Euler SSPRK(2,5) and BDF2 DeC. The results are shown in Figures 2.7 and 2.8.

On the upper surface of the airfoil, counter-clockwise vortices are observed with both temporal schemes. The initial formation of this bubble can be observed and is moved on the airfoil upper surface towards the leading edge as angle of attack increases. Once the maximal angle reached, the flow separated at the leading edge of the airfoil reattaches at the half of the chord. The

formation of the new counterclockwise bubble causes this reattachment just behind the mid-chord. The bubble length is approximately two times smaller than the first one length and the second separation is observed about the last quarter chord location. The trailing edge vortex in counter-clockwise direction now causes once more the flow to attach at the end of the chord. This bubble is well captured in the BDF-2 scheme simulation but the vortex in the wake is more accurate in the SSPRK(2,5) scheme. We know that the SSPRK(2,5) and the BDF-2 are schemes of order 2. And with the numerical results, the two schemes for concluding have quite the same results in this 2D test case.

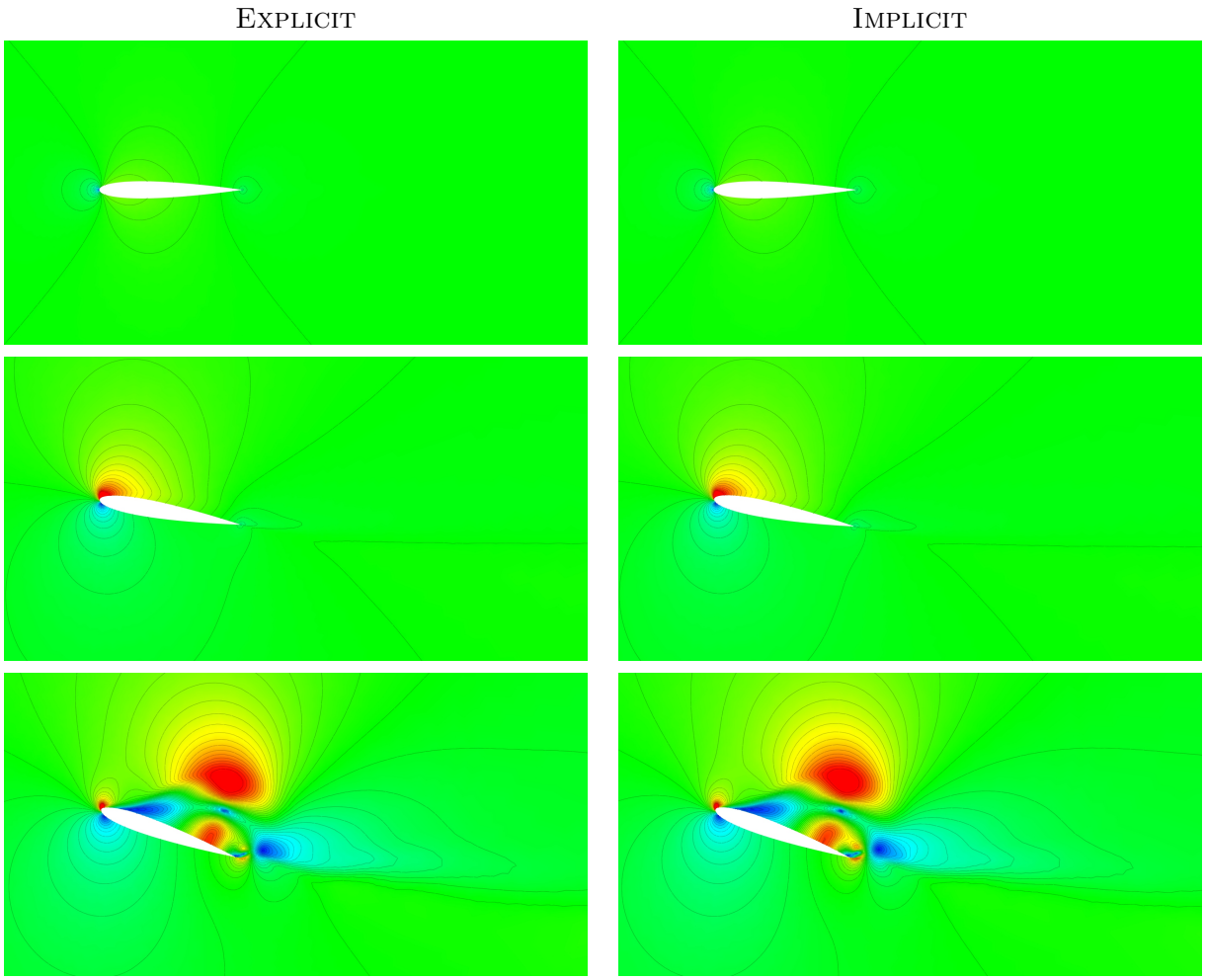


Figure 2.7: Nosing up NACA0012 test case with explicit (left) and implicit schemes (right) Mach isolines at different time steps (t = 0, 30, 45).

But the CFL for the SSPRK(2,5) explicit scheme cannot exceed 3which is the faster second-order in time explicit scheme we have, when CFL 500 is used for the BDF-2 scheme. The CPU

times of both simulations are summarized in the Table 2.2. We only divided the computation time by 2 in this two-dimensional case.

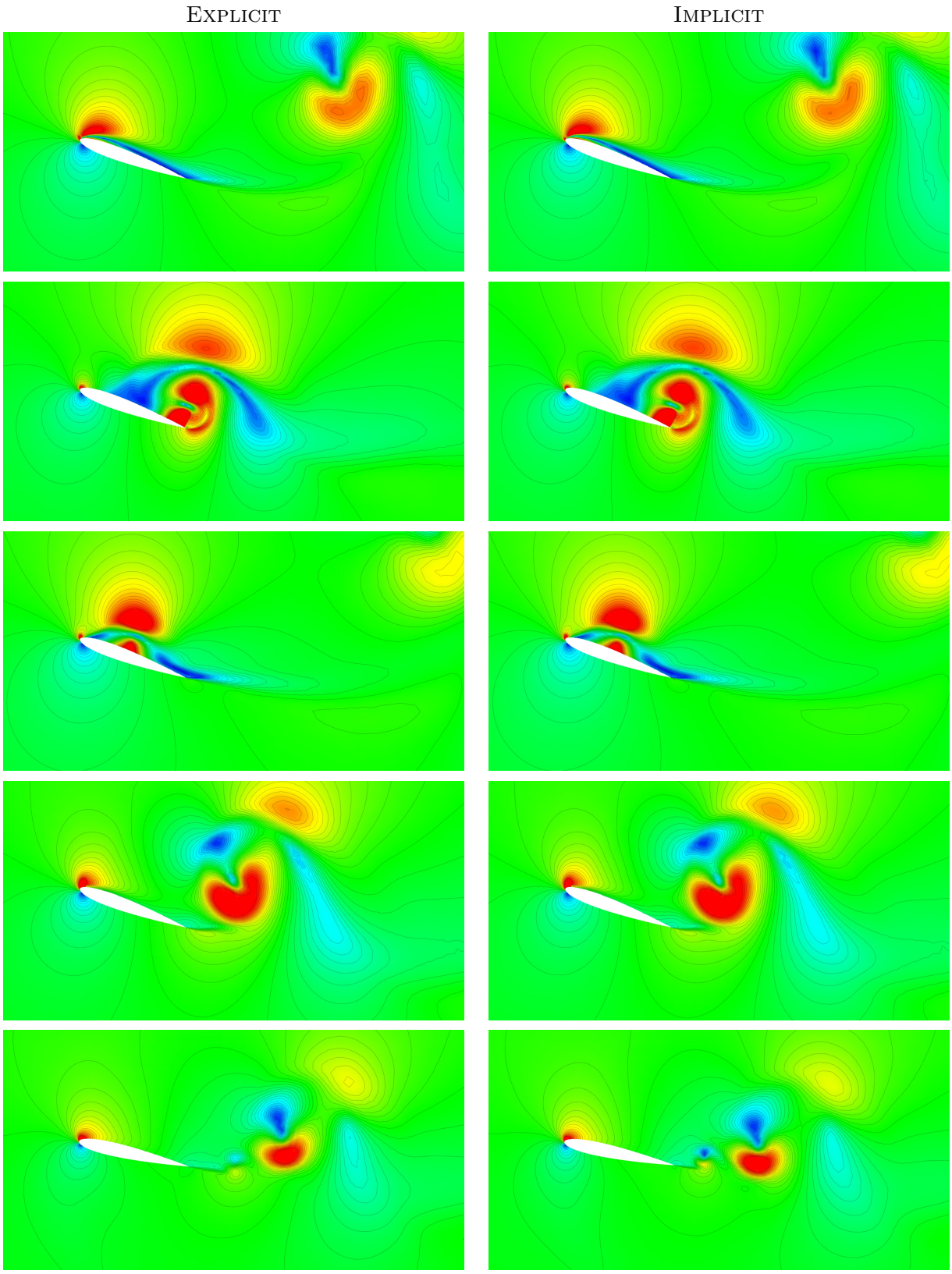


Figure 2.8: Nosing up NACA0012 test case with explicit (left) and implicit schemes (right) Mach isolines at different time steps (t = 60, 75, 90, 105, 147).

	RK2	BDF2
CPU Time	55min42s	29min24s

Table 2.2: Nosing up NACA0012 test case. Final CPU time.

2.4.2 FSI Case : 2D Bomb drop

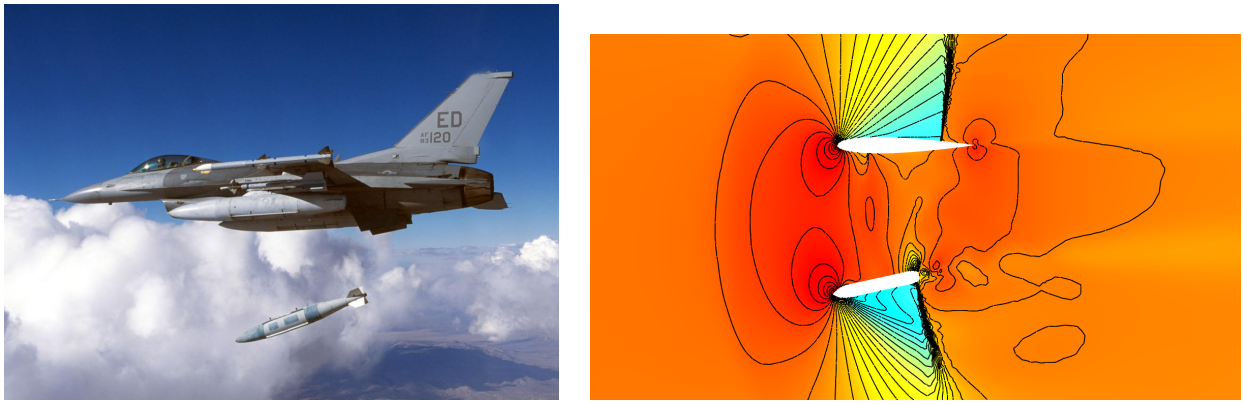


Figure 2.9: B61-12LEP dropping from a F16 (left). Corresponding 2D-computation (right)

Mesh

To compute this case, we first define a 2D mesh as a cross-section view of an initial 3D mesh represented in Figure 2.10. We first keep only the triangles involved in the intersection of the initial mesh surface and the plane $z = 3.2$. Then for each of these triangles, we compute some points at the intersection and necessarily the points at the extremities. As the wing profile looks like a NACA0012, we decide to use it instead of my section view and to keep the ratios considering the NACA0012 is 1-length. Finally we choose a circular domain of radius 20 and center $(0.5, 0)$. The final 2D mesh (see Figure 2.11) has 25779 vertices. It is generated by AFLR [Marcum 2001]. AFLR is an unstructured mesh generation software based on an advancing-front/local-reconnection method.

Initial conditions

- *Aerodynamic conditions* : Mach number $= 0.8$, Angle of Attack $= 2°$
- *External forces* To eject the bomb an ejection force is applied to provide it to move up due to the external forces and strike the aircraft. It's supposed to be $F = (0.; -0.02469)$.
- *Gravity vector* Considering the aircraft is flying below 20 km above ground so the gravity vector stands for $g = (0.; -9.74)$.

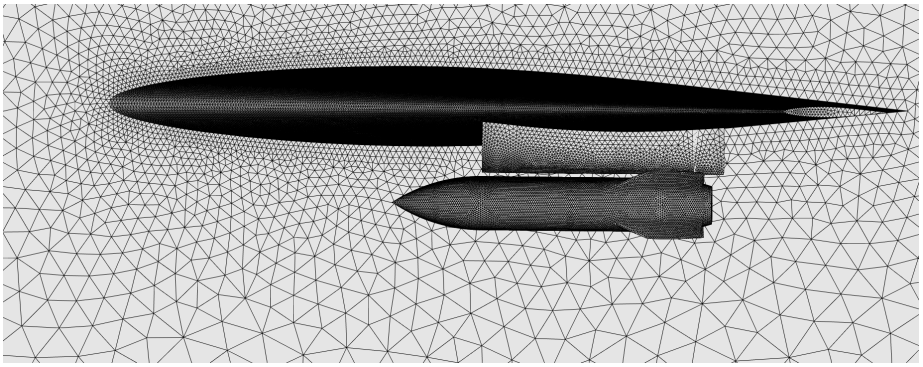


Figure 2.10: Initial 3D Mesh of the wing profile with attached bomb.

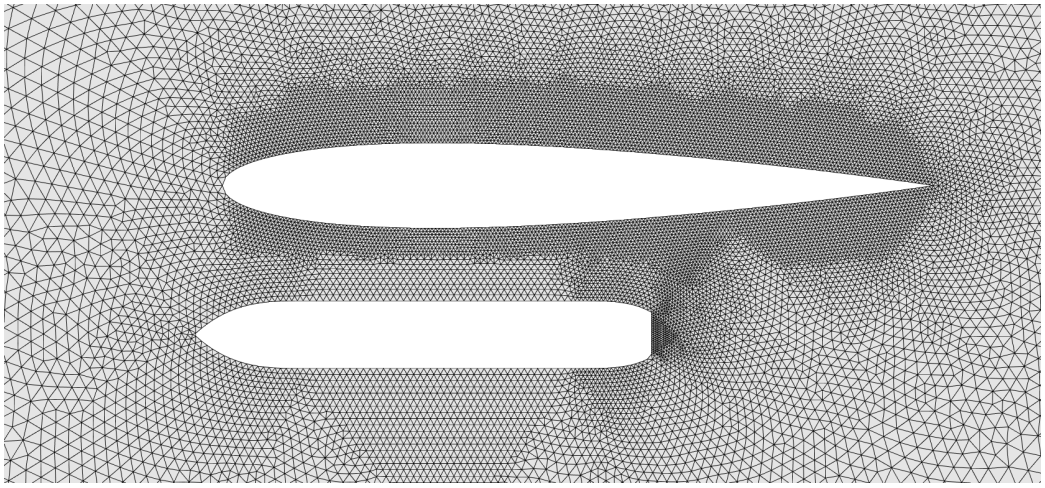


Figure 2.11: 2D Mesh : naca0012 and sectional view of the bomb.

- *Bomb charcteristics* Obviously, its gravity center must be located in the head of the bomb to fall. We consider it at $(0.06, -0.21)$. To be realistic, its mass is 500 kgs and its matrix of inertia is defined as for a cylinder of radius 0.032 and height 0.6 *i.e.* :

$$\begin{pmatrix} 15.125 & 0 & 0 \\ 0 & 15.125 & 0 \\ 0 & 0 & 0.251 \end{pmatrix}$$

Numerical results

The simulation is done with a BDF-2 scheme with DeC in FSI motion.

Observations and Interpretations

At initial time the plane is flying and carrying the bomb. At the moment the bomb is released, it has the same velocity as the aircraft. At time $t = 0$, an attached oblique shockwave is located

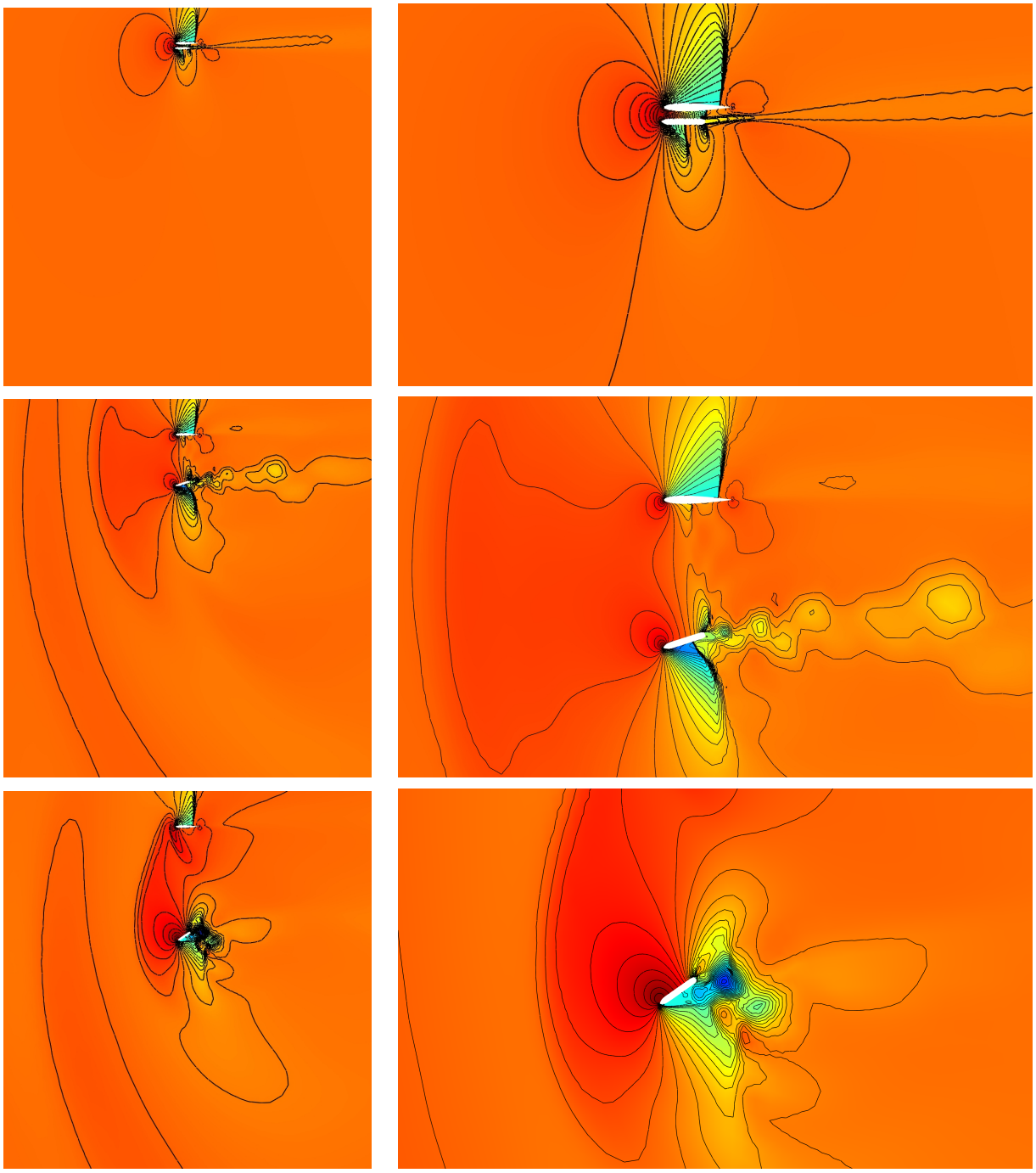


Figure 2.12: Results of the bomb drop computation at different times (left). Corresponding zoom-in (right).

at the middle of the bomb on its lower face. Vortices also appear at the tail of the bomb. When the bomb falls, this shockwave moves forward. And when the bomb is released it begins to

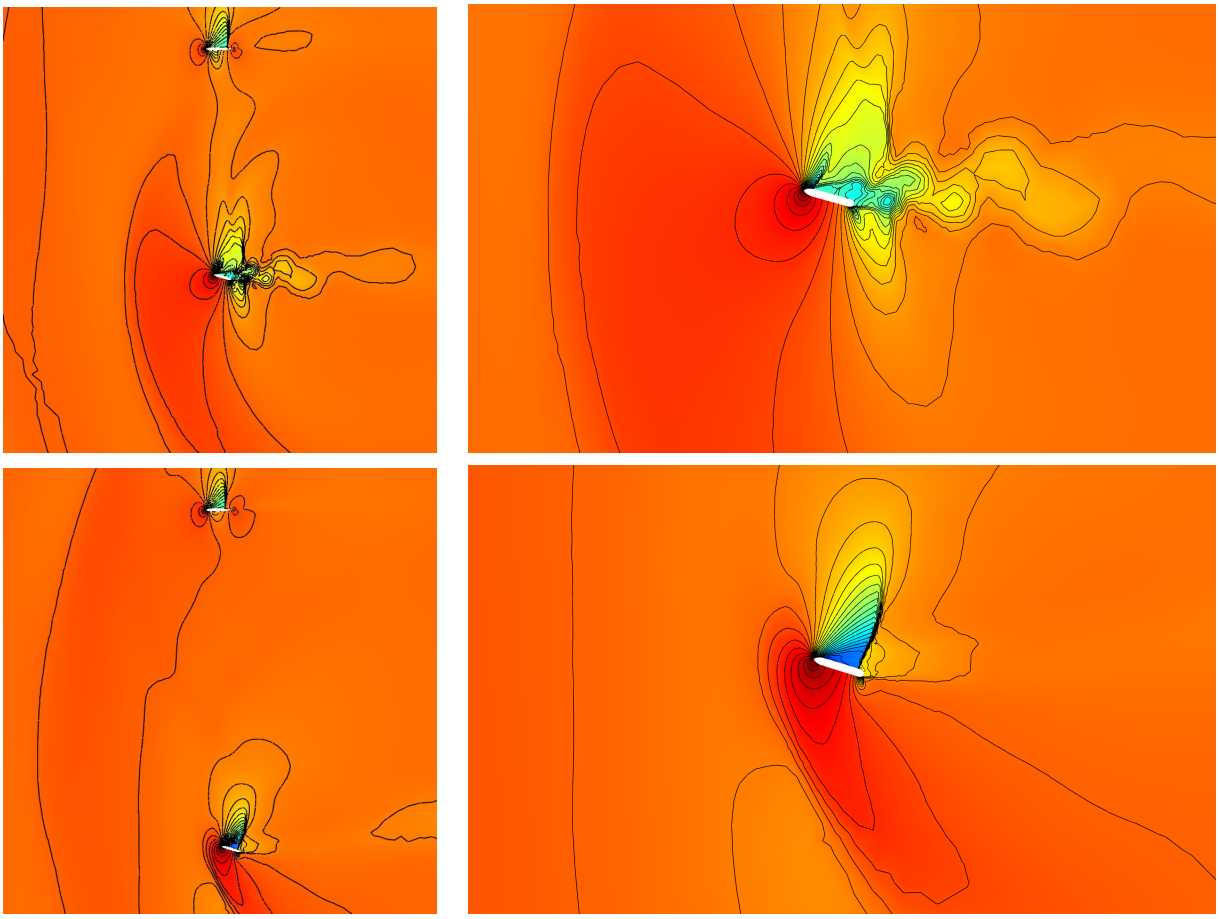


Figure 2.13: Results of the bomb drop computation at different times (left). Corresponding zoom-in (right).

accelerate down due to the gravity but it also continues the forward motion. That explains the position of the bomb below the aircraft. An other shockwave appears on the upper face of the bomb moving on the opposite way than the first one. This creates a light swing of the bomb during the drop. Results are shown in Figures 2.12 and 2.13.

2.4.3 3D Nosing-up F117

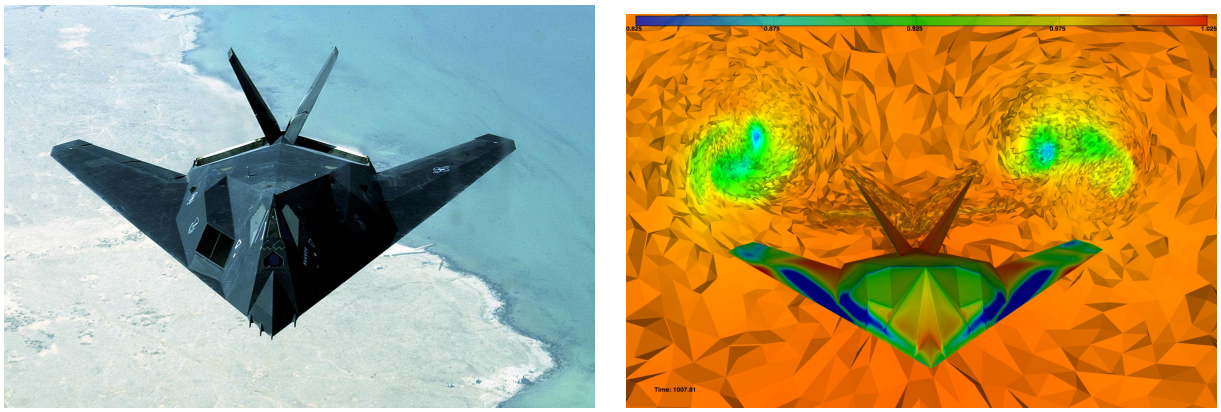


Figure 2.14: F117 flight (left). Corresponding 3D-computation (right).

The chosen case concerns vortex flow created by the nosing up/nosing down of a subsonic F-117 aircraft. As a first step the F-117 aircraft geometry nosing up, that creates a vortical wake. An inflow of air at Mach 0.4 arrives in front of the aircraft, initially in horizontal position, that noses up, stays up for a while, then noses down. In this example, the aircraft rotates around its center of gravity. Let $T = 1$ be the characteristic time of the movement and $\theta^{max} = 20^o$ the maximal angle reached, the movement is defined by its angle of rotation, of which the evolution is divided in 7 phases:

$$\theta(t) = \theta^{max} \begin{cases} 0 & \text{if} \quad 0 \leq t \leq T/2 \qquad (i) \\ \frac{2(t-\frac{T}{2})^2}{T^2} & \text{if} \quad \frac{T}{2} < t \leq T \qquad (ii) \\ \frac{1}{2} + 2\,(\frac{2\,(t-\frac{T}{2})}{T} - 1) - \frac{1}{2}\,(\frac{4\,(t-\frac{T}{2})^2}{T^2} - 1) & \text{if} \quad T < t \leq \frac{3T}{2} \qquad (iii) \\ 1 & \text{if} \quad \frac{3T}{2} < t \leq \frac{7T}{2} \qquad (iv) \\ 1 - \frac{2\,(t-\frac{7T}{2})^2}{T^2} & \text{if} \quad \frac{7T}{2} < t \leq 4T \qquad (v) \\ \frac{1}{2} - 2\,(\frac{2\,(t-\frac{7T}{2})}{T} - 1) + \frac{1}{2}\,(\frac{4\,(t-\frac{7T}{2})^2}{T^2} - 1) & \text{if} \quad 4T < t \leq \frac{9T}{2} \qquad (vi) \\ 0 & \text{if} \quad \frac{9T}{2} t \leq 5T \qquad (vii) \end{cases}$$

Phase (i) is an initialization phase, during which the flow around the aircraft is established. Phases (ii) and (iii) are respectively phases of accelerated and decelerated ascension. Vortices start to grow behind the aircraft, and they expand during phase (iv), where the aircraft stays in upward position. Phases (v) and (vi) are phases of accelerated and decelerated descent, the vortices start to move away and they slowly disappear in phase (vii). Free-stream conditions are imposed on the faces of the surrounding box, and slipping conditions on the aircraft.

Comparison between implicit and explicit solvers

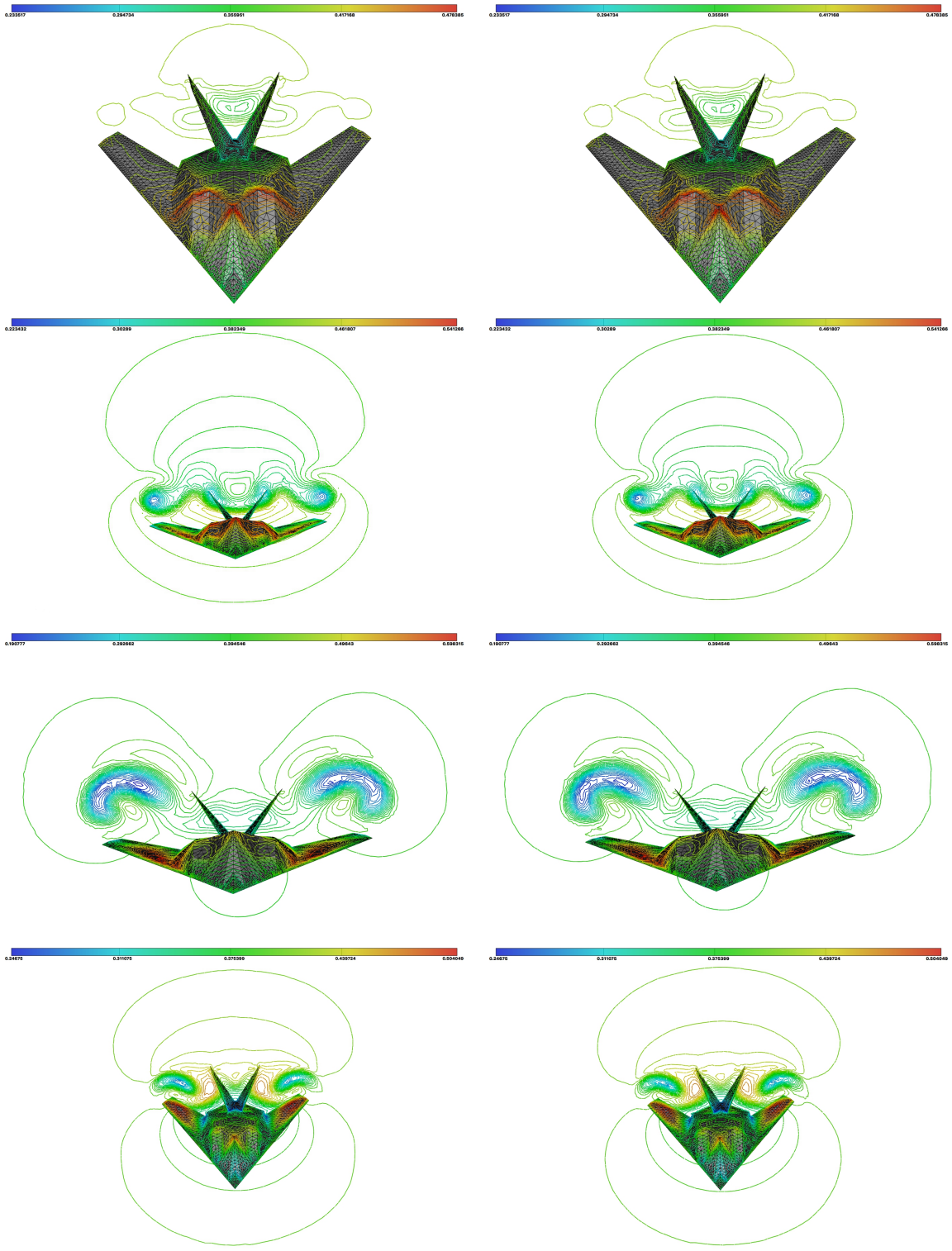


Figure 2.15: Mach isolines of nosing up f117 test case in SSPRK2 explicit scheme (left)/ implicit BDF2 scheme with defect correction (right) at different time steps [(view from the top) t = 0, 100, 200, 300].

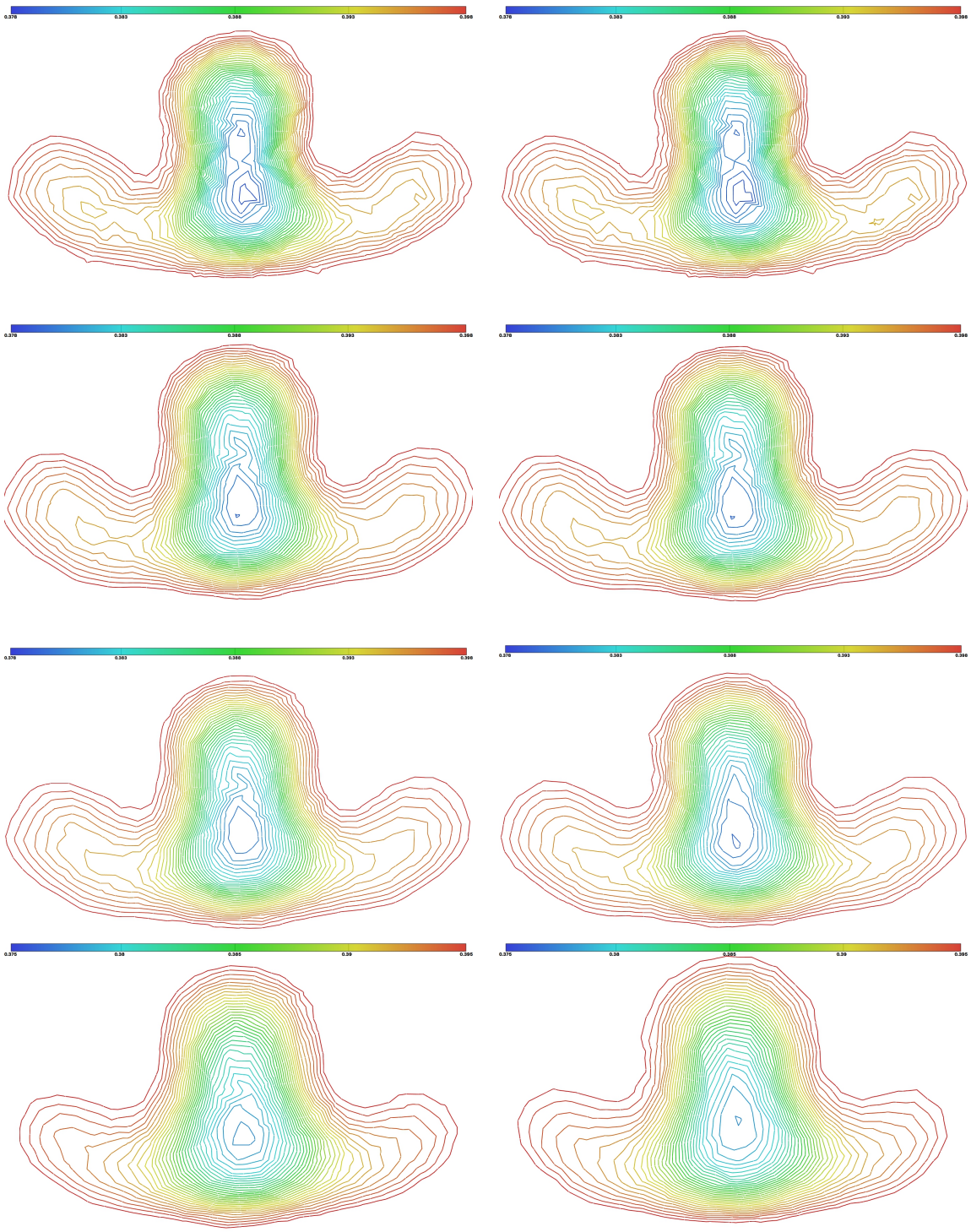


Figure 2.16: Mach isolines of nosing up f117 test case in SSPRK2 explicit scheme (left)/ implicit BDF2 scheme with defect correction (right) at different time steps [(view from the top) t = 0, 100, 200, 300].

The simulation was computed in SSPRK2, BDF-1 and BDF-2 schemes. The results are shown in SSPRK2 and BDF-2 in the Figure 3.6. As in 2D, they are pretty closed for the schemes of order 2. But the CPU time is significantly different due to the larger CFL used for the implicit time integration schemes. CFL 1 is used for SSPRK2, CFL 500 is used for the two BDF schemes.
Table of CPU time for the three simulations 2.3. The first column stands for the explicit RK2 scheme computation, the second for the BDF1 scheme, the third for the BDF2 scheme :

	SSPRK2	BDF1	BDF2
CPU Time	3h2min13s	0h39min38s	0h44min8s

Table 2.3: Nosing up F117 test case. Final CPU time.

The difference of time between BDF1 and BDF2 can be explained by the additional step computed for the second order scheme.

2.5 Conclusion

In this chapter, we presented the 3D ALE solver for the compressible Euler equations that was implemented in two and three dimensions. As regards time discretization, the multi-step Runge-Kutta schemes were already implemented in [Barral 2015] and the contribution of this thesis is the implementation of implicit methods. Details on the computation of the geometric parameters are provided. A 6-DOF mechanical model for rigid bodies was considered to add FSI coupling to the fluid solver. This solver was implemented in 3D in the same code as the moving mesh algorithm, and was plugged into it as explained in Algorithm 5. We also present numerous cases using this ALE solver and the connectivity-change moving mesh method.

In this chapter, we have seen the two aspects of simulations with moving boundaries: the handling of the moving mesh, and its consequences on the solver side. This algorithm allows us to run large displacement complex moving mesh simulations without remeshing.

Following from [Barral 2015], a novelty was presented in this chapter :

- A new integration in time was introduced. Three implicit time integration schemes ALE solver were implemented : BDF1, BDF2, Crank Nicolson schemes and parallelized in 3D. The formulas for the computation of the geometrical parameters were clarified.

- The two BDF implicit solver were validated on test cases in 2D and 3D.

- ALE simulations were run in 2D and 3D, with imposed motion or Fluid-Structure Interaction.
- These considered implicit solvers are efficient to speed-up low-mach simulations, in particular, in the presented test cases, the subsonic aeronautic cases.

Perspectives arising from this work concern both the moving mesh aspect and the solver side.

- A strategy for moving boundary layer meshes around deformable geometries naturally arises from what is presented. It needs to be developed and applied on various cases.
- More generally, handling boundary layer meshes would make it possible to run Navier-Stokes simulations.

Now, to improve the accuracy of these moving mesh simulations without increasing unreasonably their cost, we use metric-based mesh adaptation. We will see it in the next chapter.

Chapter 3

Hessian-based mesh adaptation for moving geometries

Simulating complex moving geometries evolving in unsteady flows in three dimensions is more and more required by industry. But it still remains a challenge. The first reason of this challenge is the modelling as we already seen but this is not the only explanation. Simulating moving geometries is also very costful. In fact preserving a good accuracy is often synonym of fine mesh and great number of vertices. To increase the accuracy of these simulations while preserving the same number of vertices and a similar CPU time, anisotropic metric-based mesh adaptation is one solution. Generating an adapted mesh means optimizing the accuracy of some parts of the simulation domain while de-emphazing other parts we don't need precision. Mesh adaptation allows computation of a sufficiently precise solution without enormous requirements for memory and CPU time. Anisotropic mesh adaptation is one of these mesh adaptations and is based on the control of the interpolation error of a piecewise polynomial interpolation of the exact solution on the triangular grid. The mesh adaptation criterion checks the behaviour of the second order derivatives of the solution of the considered problem. Even if in this thesis we only use it with the solver previously developed in Chapter 2, we can mention that the additional advantage of it is that it can be used without any modification for arbitrary boundary value problem and arbitrary numerical method (finite element method, finite volume method, discontinuous Galerkin method, etc.). Anisotropic mesh adaptation has already proved its efficiency for steady problems, and appears as a good perspective for unsteady problems. However, its extension to the unsteady case involving moving geometries is absolutely not straightforward. These simulations combine the difficulties arising from unsteadiness and geometrical complexity : global time step driven by the mesh smallest altitude, evolution of the phenomena in the whole domain, interpolation spoiling, but also three-dimensional meshing and remeshing issues with an imposed discretized surface. In fact, the introduction of moving geometries in this process raises new difficulties : handling of the mesh movement without deterioration of its quality, handling the specific numerical schemes imposed by moving mesh schemes and their restrictions, as well as fluid/structure coupling.

Three different approaches dealing with time-dependent mesh adaptation in the literature can

be distinguished : re-meshing methods, fixed-point algorithms and adaptative moving meshes. First, an isotropic mesh is adapted frequently in order to maintain the solution within refined regions and a safety area around critical regions is introduced [Löhner 1990, Löhner 1992, Rausch 1992, Speares 1997]. Another approach is to use an unsteady mesh adaptation algorithm [de Sampaio 1993, Wu 1990] based on the estimation of the error every n flow solver iterations and global remeshing techniques. If the error is greater than a prescribed threshold, the mesh is re-adapted. A similar strategy is adopted in [Picasso 2003, Picasso 2009], where, at each re-adaptation step, a nearly optimal metric is found by scaling the metrics in every directions depending on directional error factors. More recently, local adaptive remeshing enabling the construction of anisotropic meshes has been considered. In this case [Pain 2001, Remacle 2005, Compère 2008], the mesh is frequently adapted in order to guarantee that the solution always evolves in refined regions. All these approaches involve a large number of adaptations while introducing unquantified errors due to the transfer of the solution from the old mesh to the new one. Moreover, none of them considers the inherent non-linear nature of the mesh adaptation problem: the convergence of the mesh adaptation process is never addressed and therefore obtaining the optimal mesh cannot be expected.

A first answer to these issues has already been proposed in [Alauzet 2007]. It relies on the assumption that the temporal error is always controlled by the spatial error, which is indeed the case when solving a linear advection problem under a CFL condition. A fixed-point algorithm is used to predict the solution and converge the couple mesh/solution. The simulation interval is split into fixed-size sub-intervals on which an unique adapted mesh is used. More recently, mesh adaptation and adaptive time-stepping methods were combined for incompressible unsteady simulations [Coupez 2013]. In that approach, moving boundaries are handled via level-set methods.

In adaptive moving meshes, also called r-adaptation, the mesh is moved at each time step into a mesh adapted to the solution via a mesh equation. A large range of methods share this common principle, the differences between each lying in the adaptation criterion, the way it influences the mesh movement and the way the mesh movement is taken into account reciprocally in the physical equations. In general, they strongly couple the mesh movement and the physical equation being solved, resulting in methods with no spoiling interpolation errors. Some methods are velocity-based, meaning the mesh position is found through the vertices velocity. They include Lagrangian methods, where the velocity of the mesh is the actual Lagrangian velocity of the fluid [Loubère 2010], Moving Finite Elements, where the mesh celerity is the solution of a minimization problem [Miller 1981, Baines 1994], and methods based on the conservation of the Geometric Conservation Law [Cao 2002]. These methods tend to quickly produce tangled meshes. In location-based methods, the mesh equation directly gives the position of the vertices. The adapted meshes are seen

as images of a reference mesh by a coordinate transformation that is the solution of a minimization problem. This problem usually involves a monitor function to specify the sizes of the meshes, which is either heuristic or derived from error estimates. The problems can be based on Laplace [Winslow 1963] or Poisson [Thompson 1983] equations, or harmonic maps [Dvinsky 1991]. The application of the equidistribution principle leads to so called Moving-Mesh PDEs (MMPDEs) [Huang 2010b]. More recently, Monge-Ampère equation was also used to determine the mesh movement [Chacón 2011, Browne 2014]. These approaches seem very time-consuming for now, since the PDEs are solved at every solver time step. Moreover, the solution of the Monge-Ampère equation is known to be very difficult in three dimensions.

This section is the continuation of the work of [Olivier 2011a], [Barral 2015], [N. Barral 2017]. In those works, the authors proposed a new error estimation for simulations with moving geometries and a new ALE metric. But, mesh optimizations were not adapted consistently during the process. The contribution made in this thesis is the maintain of a consistent ALE metric in time-accurate anisotropic mesh adaptation for time-dependent problems with moving geometries.

This chapter is divided into two axes. The first sections detail the theory of error estimation and optimal metric for unsteady simulations and the numerical implementation in practice. The last sections focus on the update for dynamic meshes adaptation and the numerical choices made.

3.1 Error estimate for unsteady simulations

Let us first detail the error analysis that leads to the adaptation algorithm for unsteady simulations. This analysis is performed in the context of the continuous mesh model: first, an error model is proposed, then this error is minimized to derive an optimal mesh.

In the context of time-dependent problems, the error analysis for the steady case recalled in Section 1.2.1 is not sufficient, since it controls only spatial errors, whereas temporal errors need to be addressed too. In this work, we do not account for time discretization errors but we focus on a space-time analysis of the spatial error. In other words, we seek for the optimal space-time mesh controlling the space-time spatial discretization error. This assumption makes sense for the type of simulations that are considered in this work considering that *as an explicit time scheme is used for time advancing, then the error in time is controlled by the error in space under CFL condition.* This has been demonstrated under specific conditions in [Alauzet 2007]. As long as this hypothesis holds, the spatial interpolation error provides a fair measure of the

total space-time error of the discretized unsteady system. But it will be no longer valid in the case of future implicit time advancing solvers. The analysis would have to be completed to take into account the temporal discretization error.

3.1.1 Error model

Our goal is to solve an unsteady PDE which is set in the computational space-time domain $\mathcal{Q} = \Omega \times [0,T]$ where T is the (positive) maximal time and $\Omega \subset \mathbb{R}^3$ is the spatial domain. Let Π_h be the usual P^1 projector, we extend it to time-dependent functions:

$$\left(\Pi_h \varphi\right)(t) = \Pi_h\left(\varphi(t)\right), \ \forall\, t \in \ [0,T]\,. \tag{3.1}$$

The considered problem of mesh adaptation consists in finding the space-time mesh $\mathcal{H}$ of Ω that minimizes the space-time linear interpolation error $u - \Pi_h u$ in L^p norm. The problem is thus stated in an a priori way:

$$\text{Find } \mathcal{H}_{opt} \text{ having } N_{st} \text{ vertices such that } \ \mathbf{E}_{L^p}(\mathcal{H}_{opt}) = \min_{\mathcal{H}} \|u - \Pi_h u\|_{L^p(\Omega_h \times [0,T])}\,. \tag{3.2}$$

This problem is ill-posed and has far too many unknowns to be solved directly, as was explained previously. So it is rewritten in the continuous mesh framework under its continuous form:

$$\text{Find } \mathbf{M}_{L^p} = \left(\mathcal{M}_{L^p}(\mathbf{x},t)\right)_{(\mathbf{x},t)\in\mathcal{Q}} \ \text{ such that } \quad \mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \min_{\mathbf{M}} \|u - \pi_{\mathcal{M}} u\|_{L^p(\Omega\times[0,T])}\,, \tag{3.3}$$

under the space-time constraint:

$$\mathcal{C}_{st}(\mathbf{M}) = \int_0^T \tau(t)^{-1} \left(\int_\Omega d_{\mathcal{M}}(\mathbf{x},t)\ \mathrm{d}\mathbf{x} \right)\ \mathrm{d}t = \mathcal{N}_{st}\,. \tag{3.4}$$

where $\tau(t)$ is the time step used at time t of interval $[0,T]$. Introducing the continuous interpolation error, we recall that we can write the continuous error model as follows:

$$\mathbf{E}_{L^p}(\mathbf{M}) = \left(\int_0^T \int_\Omega \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) |H_u(\mathbf{x},t)| \mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t) \right)^p \ \mathrm{d}\mathbf{x}\ \mathrm{d}t \right)^{\frac{1}{p}}\,. \tag{3.5}$$

where H_u is the Hessian of sensor u. To find the optimal space-time continuous mesh, Problem (3.3-3.4) is solved in two steps:

(i) First, a spatial minimization is done for a fixed t.

(ii) Second, a temporal minimization is performed.

Note that both minimizations are performed formally.

3.1.2 Spatial minimization for a fixed t

Let us assume that at time t, we seek for the optimal continuous mesh $\mathbf{M}_{L^p}(t)$ which minimizes the instantaneous error, *i.e.,* the spatial error for a fixed time t:

$$\tilde{\mathbf{E}}_{L^p}(\mathbf{M}(t)) = \int_\Omega \text{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t)\,|H_u(\mathbf{x},t)|\,\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t)\right)^p \,\mathrm{d}\mathbf{x}$$

under the constraint that the number of vertices is prescribed to

$$\mathcal{C}(\mathbf{M}(t)) = \int_\Omega d_{\mathcal{M}(t)}(\mathbf{x},t)\,\mathrm{d}\mathbf{x} = \mathcal{N}(t). \tag{3.6}$$

Solving the optimality conditions provides the *optimal instantaneous continuous mesh in L^p-norm* $\mathbf{M}_{L^p}(t) = (\mathcal{M}_{L^p}(\mathbf{x},t))_{\mathbf{x}\in\Omega}$ at time t defined by:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}(t)^{\frac{2}{3}}\,\mathcal{M}_{L^p,1}(\mathbf{x},t)\,, \tag{3.7}$$

where $\mathcal{M}_{L^p,1}$ is the optimum for $\mathcal{C}(\mathbf{M}(t)) = 1$:

$$\mathcal{M}_{L^p,1}(\mathbf{x},t) = \left(\int_\Omega (\det|H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}}\mathrm{d}\bar{\mathbf{x}}\right)^{-\frac{2}{3}} (\det|H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}}\;|H_u(\mathbf{x},t)|. \tag{3.8}$$

The corresponding optimal instantaneous error at time t writes:

$$\tilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}(t)) = 3^p\,\mathcal{N}(t)^{-\frac{2p}{3}}\left(\int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}}\mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3}} = 3^p\,\mathcal{N}(t)^{-\frac{2p}{3}}\,\mathcal{K}(t)\,. \tag{3.9}$$

Throughout this thesis we denote: $\mathcal{K}(t) = \left(\int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}}\mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3}}$.

3.1.3 Temporal minimization

To complete the resolution of optimization Problem (3.3-3.4), a temporal minimization is performed in order to get the optimal space-time continuous mesh. In other words, we need to find the optimal time law $t \to \mathcal{N}(t)$ for the instantaneous mesh size. In this work, we consider the case where the time step τ is specified by the user as a function of time $t \to \tau(t)$. The analysis can be extended to the case of an explicit time advancing solver subject to Courant time step condition, but the resulting optimal mesh is too complex to be used in practice.

Let the time step τ be specified by a function of time $t \to \tau(t)$. After the spatial optimization, the space-time error writes:

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = \left(\int_0^T \tilde{\mathbf{E}}_{L^p}(\mathbf{M}_{L^p}(t))\,\mathrm{d}t\right)^{\frac{1}{p}} = 3\left(\int_0^T \mathcal{N}(t)^{-\frac{2p}{3}}\,\mathcal{K}(t)\,\mathrm{d}t\right)^{\frac{1}{p}} \tag{3.10}$$

and we aim at minimizing it under the following space-time complexity constraint:

$$\int_0^T \tau(t)^{-1}\mathcal{N}(t)\,\mathrm{d}t = \mathcal{N}_{st}. \tag{3.11}$$

We want to find the optimal distribution of $\mathcal{N}(t)$ when the space-time total number of nodes $\mathcal{N}st$ is prescribed. We can apply the following one-to-one change of variables:

$$\tilde{\mathcal{N}}(t) = \mathcal{N}(t)\tau(t)^{-1} \quad \text{and} \quad \tilde{\mathcal{K}}(t) = \tau(t)^{-\frac{2p}{3}}\,\mathcal{K}(t)\,.$$

Then, the temporal optimization problem becomes:

$$\min_{\mathbf{M}} \ \mathbf{E}_{L^p}(\mathbf{M})^p = 3^p \int_0^T \tilde{\mathcal{N}}(t)^{-\frac{2p}{3}}\,\tilde{\mathcal{K}}(t)\,\mathrm{d}t \quad \text{under constraint} \quad \int_0^T \tilde{\mathcal{N}}(t)\,\mathrm{d}t = \mathcal{N}_{st}\,.$$

The solution of this problem is given by:

$$\tilde{\mathcal{N}}_{opt}(t)^{-\frac{2p+3}{3}}\,\tilde{\mathcal{K}}(t) = const \quad \Rightarrow \quad \mathcal{N}_{opt}(t) = C(\mathcal{N}_{st})\ (\tau(t)\,\mathcal{K}(t))^{\frac{3}{2p+3}}\,.$$

Here, constant $C(\mathcal{N}_{st})$ can be obtained by introducing the above expression in space-time complexity Constraint (3.11), leading to:

$$C(\mathcal{N}_{st}) = \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\,\mathcal{K}(t)^{\frac{3}{2p+3}}\mathrm{d}t\right)^{-1}\mathcal{N}_{st}\,,$$

which completes the description of the optimal space-time metric for a prescribed time step. Using Relations (3.7) and (3.8), the analytic expression of the optimal space-time metric in L^p-norm $\mathbf{M}_{L^p}$ writes:

$$\mathcal{M}_{L^p}(\mathbf{x},t) = \mathcal{N}_{st}^{\frac{2}{3}}\left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\left(\int_\Omega (\det|H_u(\bar{\mathbf{x}},t)|)^{\frac{p}{2p+3}}\mathrm{d}\bar{\mathbf{x}}\right)\mathrm{d}t\right)^{-\frac{2}{3}}\tau(t)^{\frac{2}{2p+3}}(\det|H_u(\mathbf{x},t)|)^{-\frac{1}{2p+3}}|H_u(\mathbf{x},t)|\,. \tag{3.12}$$

We get the following optimal error:

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\mathcal{N}_{st}^{-\frac{2}{3}}\left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}}\left(\int_\Omega (\det|H_u(\mathbf{x},t)|)^{\frac{p}{2p+3}}\mathrm{d}\mathbf{x}\right)\mathrm{d}t\right)^{\frac{2p+3}{3p}}\,. \tag{3.13}$$

3.1.4 Error analysis for time sub-intervals

The previous analysis provides the optimal size of the adapted meshes for each time level. Hence, this analysis requires the mesh to be adapted at each flow solver time step. In practice this approach involves a very large number of remeshings which is very CPU consuming and spoils solution accuracy due to many solution transfers. In consequence, an adaptive strategy has been proposed in [Alauzet 2007, Alauzet 2011b] where the number of remeshings is controlled (thus drastically reduced) by considering a coarse adapted discretization of the time axis, and generating adapted meshes for several solver time steps.

The idea is to split the simulation time interval into n_{adap} *sub-intervals* $[t_{i-1}, t_i]$ for $i = 1, .., n_{adap}$. Each spatial mesh $\mathbf{M}^i$ is then kept constant during each sub-interval $[t_{i-1}, t_i]$. We could consider this partition as a *time discretization of the mesh adaptation problem*. In other

words, the number of nodes N^i of the i^{th} adapted mesh $\mathbf{M}^i$ on sub-interval $[t_{i-1}, t_i]$ should for example be taken equal to:

$$N^i = \frac{\int_{t_{i-1}}^{t_i} N_{opt}(t)\tau(t)^{-1}\mathrm{d}t}{\int_{t_{i-1}}^{t_i} \tau(t)^{-1}\mathrm{d}t}.$$

Here, we propose a different option in which we get an optimal discrete answer.

Spatial minimization on a sub-interval. Given the continuous mesh complexity $\mathcal{N}^i$ for the single adapted mesh used during time sub-interval $[t_{i-1}, t_i]$, we seek for the optimal continuous mesh $\mathbf{M}^i_{L^p}$ solution of the following problem:

$$\min_{\mathbf{M}^i} \mathbf{E}^i_{L^p}(\mathbf{M}^i) = \int_\Omega \text{trace}\left((\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\,\mathbf{H}^i_u(\mathbf{x})\,(\mathcal{M}^i)^{-\frac{1}{2}}(\mathbf{x})\right)^p \mathrm{d}\mathbf{x} \quad \text{such that} \quad \mathcal{C}(\mathbf{M}^i) = \mathcal{N}^i\,, \tag{3.14}$$

where matrix $\mathbf{H}^i_u$ on the sub-interval can be defined by either using an L^1 or an L^∞ norm:

$$\mathbf{H}^i_{L^1}(\mathbf{x}) = \int_{t_{i-1}}^{t_i} |H_u(\mathbf{x},t)|\,\mathrm{d}t \quad \text{or} \quad \mathbf{H}^i_{L^\infty}(\mathbf{x}) = \Delta t_i \max_{t\in[t_{i-1},t_i]} |H_u(\mathbf{x},t)|\,,$$

with $\Delta t_i = t_i - t_{i-1}$. Processing as previously, we get the spatial optimality condition:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = (\mathcal{N}^i)^{\frac{2}{3}}\ \mathcal{M}^i_{L^p,1}(\mathbf{x})$$

with

$$\mathcal{M}^i_{L^p,1}(\mathbf{x}) = \left(\int_\Omega (\det \mathbf{H}^i_u(\bar{\mathbf{x}}))^{\frac{p}{2p+3}}\mathrm{d}\bar{\mathbf{x}}\right)^{-\frac{2}{3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}}\ \mathbf{H}^i_u(\mathbf{x}).$$

The corresponding optimal error $\mathbf{E}^i(\mathbf{M}^i_{L^p})$ writes:

$$\mathbf{E}^i_{L^p}(\mathbf{M}^i_{L^p}) = 3^p\,(\mathcal{N}^i)^{-\frac{2p}{3}}\left(\int_\Omega (\det \mathbf{H}^i_u(\mathbf{x}))^{\frac{p}{2p+3}}\mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3}} = 3^p\,(\mathcal{N}^i)^{-\frac{2p}{3}}\,\mathcal{K}^i\,. \tag{3.15}$$

where $\mathcal{K}^i = \left(\int_\Omega \left(\det \mathbf{H}^i_u(\mathbf{x})\right)^{\frac{p}{2p+3}}\,\mathrm{d}\mathbf{x}\right)^{\frac{2p+3}{3}}$.

To complete our analysis, we shall perform a temporal minimization. Again, we consider the case where the time step τ is specified by a function of time and.

Temporal minimization for specified τ. After the spatial minimization, the temporal optimization problem reads:

$$\min_{\mathbf{M}} \mathbf{E}_{L^p}(\mathbf{M})^p = \sum_{i=1}^{n_{adap}} \mathbf{E}^i_{L^p}(\mathbf{M}^i_{L^p}) = 3^p \sum_{i=1}^{n_{adap}} (\mathcal{N}^i)^{-\frac{2p}{3}}\,\mathcal{K}^i$$

under the constraint:

$$\sum_{i=1}^{n_{adap}} \mathcal{N}^i \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1}\mathrm{d}t\right) = \mathcal{N}_{st}\,.$$

We set the one-to-one mapping:

$$\tilde{\mathcal{N}}^i = \mathcal{N}^i \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right) \quad \text{and} \quad \tilde{\mathcal{K}}^i = \mathcal{K}^i \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{3}},$$

then the optimization problem reduces to:

$$\min_{\mathbf{M}} \sum_{i=1}^{n_{adap}} (\tilde{\mathcal{N}}^i)^{-\frac{2p}{3}} \tilde{\mathcal{K}}^i \quad \text{such that} \quad \sum_{i=1}^{n_{adap}} \tilde{\mathcal{N}}^i = \mathcal{N}_{st}\,.$$

The solution is:

$$\tilde{\mathcal{N}}^i_{opt} = \mathcal{C}(\mathcal{N}_{st})\,(\tilde{\mathcal{K}}^i)^{\frac{3}{2p+3}} \quad \text{with} \quad \mathcal{C}(\mathcal{N}_{st}) = \mathcal{N}_{st} \left(\sum_{i=1}^{n_{adap}} (\tilde{\mathcal{K}}^i)^{\frac{3}{2p+3}} \right)^{-1}$$

$$\Rightarrow \quad N^i = N_{st} \left(\sum_{i=1}^{n_{adap}} (\mathcal{K}^i)^{\frac{3}{2p+3}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+3}} \right)^{-1} (\mathcal{K}^i)^{\frac{3}{2p+3}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{-\frac{3}{2p+3}}.$$

and we deduce the following optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}}$ and error:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{3}} \left(\sum_{i=1}^{n_{adap}} (\mathcal{K}^i)^{\frac{3}{2p+3}} \Big(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \Big)^{\frac{2p}{2p+3}} \right)^{-\frac{2}{3}} \Big(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \Big)^{-\frac{2}{2p+3}} (\det \mathbf{H}^i_u(\mathbf{x}))^{-\frac{1}{2p+3}}\, \mathbf{H}^i_u(\mathbf{x})\,, \tag{3.16}$$

$$\mathbf{E}_{L^p}(\mathbf{M}_{L^p}) = 3\mathcal{N}_{st}^{-\frac{2}{3}} \left(\sum_{i=1}^{n_{adap}} (\mathcal{K}^i)^{\frac{3}{2p+3}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+3}} \right)^{\frac{2p+3}{3p}}, \tag{3.17}$$

with $\mathcal{K}^i = \left(\int_\Omega (\det \mathbf{H}^i_u(\mathbf{x}))^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x} \right)^{\frac{2p+3}{3}}$.

3.1.5 Global fixed-point mesh adaptation algorithm for unsteady simulation

Finally, the unsteady adaptation algorithm can be derived from this error analysis. Three main ideas govern this algorithm:

- It is based on splitting the simulation into sub-intervals.
- It is an iterative fixed point algorithm.
- It is a global algorithm.

The methodology consists in splitting the simulation time frame $[0,\, T]$ into n_{adap} adaptation sub-intervals:

$$[0,\, T] \;=\; [0 = t^0,\, t^1] \cup \ldots \cup [t^i,\, t^{i+1}] \cup \ldots \cup [t^{n_{adap}-1},\, t^{n_{adap}}]\,,$$

and to keep the same adapted mesh for each time sub-interval. On each sub-interval, the mesh is adapted to control the solution accuracy from t^i to t^{i+1}. Consequently, the time-dependent simulation is performed with n_{adap} different adapted meshes. This drastically reduces the number of remeshing during the simulation, hence the number of solution transfers. This can been seen as a coarse adapted discretization of the time axis, the spatial mesh being kept constant for each sub-interval when the global space-time mesh is visualized, thus providing a first answer to the adaptation of the whole space-time mesh.

We have seen in Chapter 1 that mesh adaptation is a non-linear problem, and that this is addressed with iterative algorithms to converge the mesh/solution couple. To that end we propose a fixed-point mesh adaptation algorithm. This is also a way to predict the solution evolution and to adapt the mesh accordingly.

Previously [Alauzet 2007], the optimal metric of a sub-interval could be computed directly once the simulation on the sub-interval had been run. However, the computation of the optimal continuous mesh for sub-intervals given by Relation (3.16) involves a global normalization term which requires the knowledge of quantities over the whole simulation time frame. In this case, the normalization term is:

$$N_{st}^{\frac{2}{3}} \left(\int_0^T \tau(t)^{-\frac{2p}{2p+3}} \left(\int_\Omega (\det |H_u(\bar{\mathbf{x}}, t)|)^{\frac{p}{2p+3}} \, \mathrm{d}\bar{\mathbf{x}} \right) \mathrm{d}t \right)^{-\frac{2}{3}},$$

which requires to know all the time steps $\tau(t)$ and Hessians $H_u(\mathbf{x}, t)$ over time frame $[0, T]$. Thus, the complete simulation must be performed before evaluating any continuous mesh.

Against, considering a *global* fixed-point mesh adaptation algorithm covering the whole time frame $[0, T]$ enables to compare the normalization term. All the solutions and Hessian-metrics are computed, and only then can the global normalization term and thus the metrics for each sub-interval be computed. This algorithm is schematized in Algorithm 8 where $\mathcal{H}$, $\mathcal{S}$ and $\mathcal{M}$ denote respectively meshes, solutions and metrics, and $\mathbf{H}$ is the Hessian-metric.

Algorithm 8 Mesh Adaptation Loop for Unsteady Flows

Initial mesh and solution$(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity $\mathcal{N}_{st}$

// Fixed-point loop to converge the global space-time mesh adaptation problem

For $j = 1, n_{ptfx}$

1. *// Adaptive loop to advance the solution in time on time frame* $[0, T]$
   For $i = 1, n_{adap}$

   (a) $\mathcal{S}_{0,i}^j$ = Interpolate conservatively next sub-interval initial sol. from $(\mathcal{H}_{i-1}^j, \mathcal{S}_{i-1}^j, \mathcal{H}_i^j)$;

   (b) $\mathcal{S}_i^j$ = Compute solution on sub-interval from pair $(\mathcal{S}_{0,i}^j, \mathcal{H}_i^j)$;

   (c) $|\mathbf{H}|_i^j$ = Compute sub-interval Hessian-metric from sol. sample $(\mathcal{H}_i^j, \{\mathcal{S}_i^j(k)\}_{k=1,nk})$;

   EndFor

2. $\mathcal{C}^j$ = Compute space-time complexity from all Hessian-metrics $(\{|\mathbf{H}|_i^j\}_{i=1,n_{adap}})$;

3. $\{\mathcal{M}_i^j\}_{i=1,n_{adap}}$ = Compute all sub-interval unsteady metrics $(\mathcal{C}^j, \{|H_{\max}|_i^j\}_{i=1,n_{adap}})$;

4. $\{\mathcal{H}_i^{j+1}\}_{i=1,n_{adap}}$ = Generate all sub-interval adapted meshes $(\{\mathcal{H}_i^j,\ \mathcal{M}_i^j\}_{i=1,n_{adap}})$;

EndFor

3.2 From theory to practice

3.2.1 Computation of the Hessian metric

Since Hessians are the basis of the computation of metrics, it is important to compute them with enough accuracy.

Gradient and Hessian recovery techniques

From a piecewise linear function u_h, the chosen method must be able to recover a smooth Hessian field. Two main approaches coexist: a double L^2 projection method, and a weighted least square method. The second method is detailed in [Menier 2015]. The first one is the one generally used in this thesis, and works as follows.

Nodal gradients. Let K be an element and $(P_i)_{i=0...3}$ its vertices. Let $\mathbf{x} \in K$. u_h is written:

$$u_h(\mathbf{x}) = \sum_{j=0}^{3} u_h(P_j)\, \varphi_j(\mathbf{x})\,,$$

where $(\varphi_j)_j$ are the $\mathbf{P}^1$ shape functions, given by:

$$\nabla_{\boldsymbol{x}}\varphi_0(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_0}}, \quad \nabla_{\boldsymbol{x}}\varphi_1(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_1}}, \quad \nabla_{\boldsymbol{x}}\varphi_2(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_2}}, \quad \nabla_{\boldsymbol{x}}\varphi_3(\mathbf{x}) = \frac{1}{6|K|}\overline{\boldsymbol{\eta_3}}\,.$$

The expression is differentiated term by term:

$$\nabla u_h|_K = \sum_{j=0}^{n} u_h(P_j)\,\nabla_{\mathbf{x}}\varphi_j(\mathbf{x})\,.$$

As ∇u_h is not defined at the vertices of the mesh, the nodal gradients are recovered from the gradients to the elements using a Clément's L^2 local projection.

Let P_i be a vertex of $\mathcal{H}$. The stencil of the shape function φ_i is the ball of vertices around P_i, $\mathrm{Ball}(P_i)$. The following spaces are introduced:

$$\begin{aligned} V_h^0 &= \{v \in L^2(\Omega) \;\big|\; v_{|K} \in \mathbf{P}^0 \;\; \forall K \in \mathcal{H}\} \\ V_h^1 &= \{v \in \mathcal{C}^0(\Omega) \;\big|\; v_{|K} \in \mathbf{P}^1 \;\; \forall K \in \mathcal{H}\}\,. \end{aligned}$$

where $\mathbf{P}^0$ and $\mathbf{P}^1$ are the set of polynomials of degree respectively 0 (constant polynomials) and 1.

For $v \in L^2(\Omega)$, we set $\Pi_{L^2} v \in V_h^0$:

$$\forall\,\mathrm{Ball}(P_i) \subset \mathcal{H}\,, \quad \begin{cases} (\Pi_{L^2} v)_{|\mathrm{Ball}(P_i)} \in \mathbf{P}^0 \\ \int_{\mathrm{Ball}(P_i)} (\Pi_{L^2} v - v)\; w = 0\,, \;\; \forall w \in \mathbf{P}^0\,. \end{cases}$$

Clement's operator $\Pi_c : V_h^0 \longrightarrow V_h^1$ is defined as follows:

$$\Pi_c v = \sum_{i=0}^{n} \Pi_{L^2} v(P_i)\,\varphi_i\,.$$

With Clément's operator, we can recover nodal gradients from the $\nabla u_h \in \mathbf{P}^0$. For each $\mathrm{Ball}(P_i) \subset \mathcal{H}$ we can in particular take $v = 1 \in \mathbf{P}^0$ as test function:

$$\begin{aligned} \int_{\mathrm{Ball}(P_i)} (\Pi_{L^2}(\nabla u_h) - \nabla u_h) = 0 \quad &\Longleftrightarrow \quad \int_{\mathrm{Ball}(P_i)} \Pi_{L^2}(\nabla u_h) = \int_{\mathrm{Ball}(P_i)} \nabla u_h \\ &\Longleftrightarrow \quad |\mathrm{Ball}(P_i)|\,\Pi_{L^2}(\nabla u_h)_{|\mathrm{Ball}(P_i)} = \sum_{K_j \in \mathrm{Ball}(P_i)} \int_{K_j} \nabla u_h \\ &\Longleftrightarrow \quad \Pi_{L^2}(\nabla u_h)_{|\mathrm{Ball}(P_i)} = \frac{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|\,\nabla u_{h|K_j}}{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|} \end{aligned}$$

where $|K|$ is the volume of element K . For each vertex P_i, we can write:

$$\nabla_R\, u_h(P_i) = \frac{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|\,\nabla u_{h|K_j}}{\sum\limits_{K_j \in \mathrm{Ball}(P_i)} |K_j|}$$

This procedure comes to recovering the gradient as an average of the gradients at the elements, weighted by the volume of the elements.

We thus have defined values to the vertices for the gradient, and consequently, thanks to Clément's operator, a piecewise P^1 gradient on the mesh.

Hessian recovery. To recover the Hessian matrix by means a double L^2 projection, the same procedure is applied once more, but using the gradient as input.

The optimal L^p metric involves an averaged Hessian-metric $\mathbf{H}_u^i$ on sub-interval i, but it still remains to know how to compute it practically, *i.e.*, how it is discretized. The strategy adopted [Alauzet 2007] is to sample the solution on the time sub-interval. More precisely, n_k solutions equally distributed on the sub-interval time frame are saved, including the initial solution at t^{i-1} and the final solution at t^i. Positive Hessian $|H_u(\mathbf{x}, t^k)|$ is evaluated for each sample. If samples are distributed regularly in time, the time elapsed between two samples is $\frac{\Delta t_i}{n_k - 1}$. In practice, 20 samples per sub-intervals are usually used.

Once these Hessians are computed, one need to average them. In the previous algorithm [Alauzet 2007], the following discretization was used:

$$\mathbf{H}^i_{L^\infty}(\mathbf{x}) = \Delta t_i \bigcap_{k=1}^{n_k} |H_u(\mathbf{x}, t^k)| = \Delta t_i \, |H^i_{\max}(\mathbf{x})| \,,$$

where $\cap$ has to be understand as the metric intersection in time of all samples. This corresponds to an integration in time in L^∞ norm of the Hessians.

However, the new error analysis leads to write $\mathbf{H}^i$ as the integral over time of the Hessian matrices, and thus the following discretization is preferred:

$$\mathbf{H}^i_{L^1}(\mathbf{x}) = \frac{1}{2}\frac{\Delta t_i}{n_k - 1}|H_u(\mathbf{x}, t_{i-1})| + \frac{\Delta t_i}{n_k - 1}\sum_{k=2}^{n_k-1} |H_u(\mathbf{x}, t_k)| + \frac{1}{2}\frac{\Delta t_i}{n_k - 1}|H_u(\mathbf{x}, t_i)| = \Delta t_i \, |H^i_{\text{avg}}(\mathbf{x})| \,,$$

where $\Delta t_i = t^i - t^{i-1}$ is the sub-interval time length and $t^k = t^{i-1} + \frac{k-1}{n_k - 1}\Delta t^i$. This corresponds to an integration in time in L^1 norm of the Hessians.

A comparative study of both discretizations was performed and will be detailed in section 3.2.2.

3.2.2 Choice of the mean Hessian-metric

All the unsteady adaptation algorithm relies on the computation of a mean metric for each sub-intervals. Instead of averaging real metrics, we average the Hessians on the go during the solver iterations.

There is a choice to make concerning the way this average is done, between the historical

intersection or the sum dictated by theory. The intersection of metrics reads:

$$\mathbf{H}^i_{L^\infty}(\mathbf{x}) = \Delta t_i \bigcap_{k=1}^{n_k} |H_u(\mathbf{x}, t^k)| = \Delta t_i \, |H^i_{\max}(\mathbf{x})| \,,$$

It comes to making an average in L^∞ norm, and has the advantage of having a clear geometric interpretation: the intersection of two metrics is the largest metric included in the two metrics. In the context of unsteady adaptation, this means that the mean Hessian-metric has the smallest sizes met over the sub-interval. The sum of metrics comes to making an average in L^1 norm, and reads:

$$\mathbf{H}^i_{L^1}(\mathbf{x}) = \frac{1}{2}\frac{\Delta t_i}{n_k - 1}|H_u(\mathbf{x}, t_{i-1})| + \frac{\Delta t_i}{n_k - 1}\sum_{k=2}^{n_k-1}|H_u(\mathbf{x}, t_k)| + \frac{1}{2}\frac{\Delta t_i}{n_k - 1}|H_u(\mathbf{x}, t_i)| = \Delta t_i \, |H^i_{\text{avg}}(\mathbf{x})| \,,$$

It seems to be more coherent with theory. Indeed, when writing the minimization problem (3.14) with sub-intervals, we can formally derive the error from the unsteady error without sub-intervals (3.5) by moving the integral over time into the trace. However, it is only a formal manipulation, and the geometrical interpretation of the sum is much less clear: a term-to-term sum of metrics is clearly still a metric (a symmetric positive definite form), but the sizes of the axes of the resulting metric are not the sum of the sizes of the two input metrics. This justifies numerical experimentation to select the best approach in practice.

3.2.3 Choice of the optimal continuous mesh

The optimal adapted mesh for each sub-interval is generated according to analysis of Section 3.1.4. For the numerical results presented below, we select the optimal mesh given by Relation (3.16) and the following particular choice has been made:

- the Hessian-metric for sub-interval i is discretized in time in L^1 norm.
- function $\tau : t \to \tau(t)$ is constant and equal to 1
- all sub-intervals have the same time length Δt.

Moreover, it is clear that integral $\int_{t_{i-1}}^{t_i} \tau(t)^{-1}\mathrm{d}t$ corresponds to the number of time steps (iterations) performed by the flow solver during the i^{th} sub-interval. In practice using the number of iterations of the flow solver for each sub-interval to define the continuous mesh is an issue because it highly depends on the discrete representation of the continuous mesh, *i.e.,* the generated discrete mesh. Thus, the number of iterations of a sub-interval may substantially varies between two fixed-point iterations. To avoid this issue, we prefer considering the flow solver time step constant on each sub-interval, and consider the time step $\tau(t)$ constant equal to Δt (because of the global normalization term). The integral $\int_{t_{i-1}}^{t_i} \tau(t)^{-1}\mathrm{d}t$ then reduces to 1.

With these choices, the optimal continuous mesh $\mathbf{M}_{L^p} = \{\mathbf{M}^i_{L^p}\}_{i=1,..,n_{adap}}$ simplifies to:

$$\mathcal{M}^i_{L^p}(\mathbf{x}) = \mathcal{N}_{st}^{\frac{2}{3}} \left(\sum_{j=1}^{n_{adap}} \left(\int_\Omega (\det |\mathbf{H}^j_{L^1}(\mathbf{x})|)^{\frac{p}{2p+3}} \mathrm{d}\mathbf{x} \right) \right)^{-\frac{2}{3}} \left(\det |\mathbf{H}^i_{L^1}(\mathbf{x})|\right)^{-\frac{1}{2p+3}} |\mathbf{H}^i_{L^1}(\mathbf{x})| \,. \quad (3.18)$$

In that case, as we assume that theoretically one time step is done by sub-interval, $\mathcal{N}_{st}/n_{adap}$ represents the average spatial complexity by sub-interval and thus $\mathcal{N}_{st}$ is the total spatial complexity by summing the sub-intervals average complexity. We do not prescribe the temporal complexity, *i.e.,* we do not control the number of time steps done at each sub-interval.

Specification of the space-time complexity. The optimal continuous mesh from Equation (3.16) is obtained for a given space-time complexity that is the number of vertices times the number of solver time steps. Such a complexity should thus be prescribed for each adaptative simulation. However, in practice, it depends on the time discretization of the sub-intervals. Moreover, as explained above, the number of solver time steps is highly dependent on the sizes of the elements of the mesh, which is not perfectly controlled throughout the algorithm. For these reasons, we prefer to prescribe an average number of vertices per sub-interval, which we sometimes refer to as "simplified space-time complexity" in what follows [Alauzet 2012].

3.2.4 Matrix-free P1-exact conservative solution transfer

For the unsteady adaptation algorithm, the interpolation step is crucial. Indeed, when moving from a sub-interval to the following, the solution has to be transferred from the mesh of the current sub-interval to the mesh of the following sub-interval, and this is done as many times as there are sub-intervals. It is obvious that if information is lost during the transfer process, then the accuracy of the global solution is greatly affected. The following properties need to be satisfied in the interpolation method: (i) mass conservation, (ii) P_1 exactness preserving the second order of the adaptive strategy and (iii) verify the maximum principle. The interpolation scheme used is detailed below.

Interpolation step

The classical approach involves two steps: first the localization of the vertices of the new mesh in the background mesh, and then the application of an interpolation scheme. The localization step is described in several references [Frey 2008, Alauzet 2010b]. As concerns interpolation schemes, the easiest is the classical P^1 interpolation, which is written:

$$u(\mathbf{p}) = \sum_{i=0}^{3} \beta_i(\mathbf{p})\, u(\mathbf{q}_i)\,, \quad (3.19)$$

where $\mathbf{p}$ is a vertex of the new mesh that has been located in tetrahedron $K = [\mathbf{q}_0, \mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3]$ of the background mesh and β_i are the barycentric coordinates of $\mathbf{p}$ with respect to K. Higher order interpolation schemes can be devised, for instance using P^2 Lagrange shape functions. However these schemes do not conserve the mass.

A new conservation scheme was devised to match these criteria [Alauzet 2010b]. The mass conservation property of the interpolation operator is achieved by local mesh intersections, *i.e.,* intersections are performed at the element level. The use of mesh intersection to build a conservative interpolation process seems natural for unconnected meshes. The idea is to find, for each element of the new mesh, its geometric intersection with all the elements of the background mesh it overlaps and to mesh this geometric intersection with simplices. We are then able to use a Gauss quadrature formula to exactly compute the mass which has been locally transferred.

High-order accuracy is obtained through the reconstruction of the gradient of the solution from the discrete data and the use of some Taylor formulas. Unfortunately, this high-order interpolation can lead to a loss of monotonicity. The maximum principle is recovered by correcting the interpolated solution in a conservative manner. Finally, the solution values at vertices are reconstructed from this piecewise linear by element discontinuous representation of the solution.

The approach is summarized in Algorithm 9:

Algorithm 9 Conservative Interpolation Process

Piecewise linear (continuous or discontinuous) representation of the solution on $\mathcal{H}_{back}$

1. For all elements $K_{back} \in \mathcal{H}_{back}$, compute solution mass $m_{K_{back}}$ and gradient $\nabla_{K_{back}}$
2. For all elements $K_{new} \in \mathcal{H}_{new}$, recover solution mass $m_{K_{new}}$ and gradient $\nabla_{K_{new}}$:
   (a) compute the intersection of K_{new} with all $K^i_{back} \in \mathcal{H}_{back}$ it overlaps
   (b) mesh the intersection polygon/polyhedron of each couple of elements (K_{new}, K^i_{back})
   (c) compute $m_{K_{new}}$ and $\nabla_{K_{new}}$ using Gauss quadrature formulas

   $\Longrightarrow$ a piecewise linear discontinuous representation of the mass on $\mathcal{H}_{new}$ is obtained
3. Correct the gradient to enforce the maximum principle
4. Set the solution values to vertices by an averaging procedure.

3.2.5 The remeshing step

The remeshing step is also crucial in the adaptation process, as a poorly adapted mesh or a mesh with bad quality elements will impact the accuracy of the solution and spoil the efforts

on all other parts of the process. In this paper, the remesher `Feflo.a` [Loseille 2013] was used. `Feflo.a` belongs to the class of 3D anisotropic local remeshers that aims at generating a unit mesh with respect to a prescribed metric field. Its main particularity is to adapt the volume and the surface mesh in a coupled way so that a valid 3D mesh is always guaranteed on output. It uses a unique cavity-based operator (that generalizes standard operators: point insertion, edge removal, edges swapping, point smoothing). This operator is governed by dedicated metric-based quality functions. This allows to reach a good balance between high level of anisotropy, necessary to capture the physical features of the solution, and mesh quality, necessary to ensure the stability and accuracy of the numerical scheme. `Feflo.a` has several enhancements designed for CFD computations, including explicit control and optimization of tetrahedra to ensure a maximal time step for unsteady simulations, and surface mesh re-projection based either on CAD or on background discrete meshes.

3.2.6 Software used

Our implementation of the mesh adaptation algorithm described above requires successively four different software components:

- `Wolf` the second order Finite-Volume flow solver, whose ALE version was described in Chapter 2 - when fixed-mesh simulations are run, a standard version is used, but the numerical schemes are the same as the ALE ones, without the extra ALE velocity terms -
- `Metrix` to compute the continuous space-time mesh and perform the metric fields gradation,
- `Feflo.a` the local adaptive remesher based on the cavity operator, described in Section 3.2.5,
- `Interpol` for the P^1-conservative solution transfer, described in Section 3.2.4.

3.3 Unsteady mesh adaptation for dynamic meshes

The strategy adopted to move the meshes is the one described in Section 2.1 : one body- fitted mesh is deformed to follow the moving boundaries, and an Arbitrary-Lagrangian-Eulerian (ALE) formulation of the equations is solved on this moving mesh. This moving mesh strategy preserves the number of vertices, which is a requirement in our error previous analysis framework. An ALE metric, which brings a first answer to this problem, was proposed in [Olivier 2011b] and was improved in [N. Barral 2017].

Section 1.3.2 provides the optimal instantaneous ALE continuous mesh which takes into account the mesh deformation. Now, as stated previously, we can extend the space-time error analysis with time sub-intervals done for fixed meshes to the case of dynamic meshes. The simulation time interval is still split into n_{adap} sub-intervals. On each sub-interval, the mesh

size (number of vertices) remains constant, but the mesh is deformed to follow the geometry displacement. At each time-step of the sub-interval, we want the moved mesh to be adapted to the current sensor. The key idea to perform the error analysis is to seek for the optimal dynamic continuous mesh at the beginning of the sub-interval, this continuous mesh being optimal for the whole sub-interval when deformed, instead of seeking for the expression of the optimal continuous mesh at each instant, *i.e.,* as a function of the time.

The optimal space-time ALE continuous mesh is designed to fit in the global fixed-point unsteady mesh adaptation algorithm described in Algorithm 10. Nevertheless, a few things have been modified to extend this algorithm to moving mesh ALE simulations. In fact, first, geometries move so mesh must be moved with all the difficulties that this implies to preserve accuracy and a good mesh quality all along the movement.

Algorithm 10 Feature-based mesh adaptation for unsteady flows with moving geometries

Input : Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity N_{st}

 # Fixed-point loop to converge the global space-time goal-oriented mesh adaptation problem
For $j = 1, n_{ptfx}$

 # Adaptive loop to advance the solution in time on time frame $[0, T]$

1. **For** $i = 0, n_{adap}$
   (a) $(\mathcal{S}_{0,i}^j)$= Interpolate conservatively next sub-interval initial solution from pair $(\mathcal{H}_{i-1}^j, \mathcal{S}_{i-1}^j, \mathcal{H}_i^j)$;
   (b) **For** $n = 0, n_{sample}$
      i. $(\mathcal{S}_i^j)$= Compute solution with the flow solver on sub-interval from pair $(\mathcal{H}_i^j, \mathcal{S}_i^0)$;
      ii. $|\mathbf{H}|_i^j$ = Compute sub-interval Hessian-metric from solution sample $(\mathcal{H}_i^j, \{\mathcal{S}_i^j(k)\}_{k=1}^{nk})$;
      iii. $\mathcal{M}_{L^p}^{ALE,n}$ = Compute ALE metric;
      iv. $\mathcal{H}_i^{ALE,n}$ = Generate adapted mesh;
      v. $\mathcal{H}_i^{ALE,n+1}$ = Move mesh using the connectivity-change algorithm;
      EndFor
   EndFor
2. $\mathcal{C}^j$ = Compute space-time complexity from all Hessian metrics $\{|\mathbf{H}|_i^j\}_{i=1}^{n_{adap}}$
3. $\{\mathcal{M}_{L^p,i}^j\}_{i=1}^{n_{adap}}$ = Compute all sub-interval metrics according to error estimate from $(\mathcal{C}^j, \{|\mathbf{H}|_i^j\}_{i=1}^{n_{adap}})$;
4. $\{\mathcal{H}_i^{j+1}\}_{i=1}^{n_{adap}}$ = Generate all sub-interval adapted meshes from pair $(\{\mathcal{H}_i^j, \mathcal{M}_{L^p,i}^j\}_{i=1}^{n_{adap}})$;

End For

3.4 Handling the swaps : update of the adaptation algorithm

The mesh optimization procedure of the connectivity-change moving mesh strategy requires a proper metric field to evaluate geometric quantities and elements qualities of anisotropic adapted meshes. But, at a given fixed-point iteration j and at a given time sub-interval i, the input is the optimal ALE continuous mesh $\mathbf{M}_{L^p}^{i,\mathrm{ALE}}$ defined on $\Omega(t^i)$. Therefore, when the mesh is deformed during the sub-interval due to the geometry displacement, the input metric is no more compatible with the moved mesh at a given time t. The use of an incorrect metric field will drive the smoothing to move vertices to a wrong location and the swaps to break the anisotropy, thus spoiling the adaptation of the mesh. As the mesh is evolving in time, the input metric field (*i.e.,* the input continuous mesh) must also evolve in time. Consequently, we have to apply the deformation of the adapted mesh to the continuous mesh in order to maintain the consistency between the discrete and the continuous meshes.

3.4.1 Metric for optimizations

But, in the theory developed in Chapter 1, the mesh deformation correction is applied to the Hessian-metric $|H_u^{i,\mathrm{ALE}}|$ and not directly to the continuous mesh. Therefore, from the optimal ALE continuous mesh $\left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t^i))\right)_{\mathbf{x}\in\Omega(t^i)}$ at the beginning of sub-interval $[t^i, t^{i+1}]$, we cannot directly find the deformed optimal ALE continuous mesh $\left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t))\right)_{\mathbf{x}\in\Omega(t)}$ for $t \in [t^i, t^{i+1}]$.

Two possibilities arise from these remarks. First, several ALE metric field samples denoted $\left\{\left(\mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}(t^k))\right)_{\mathbf{x}\in\Omega(t^k)}\right\}_k$ can be pre-calculated for each sub-interval at the previous fixed-point iteration, and when an ALE metric field is required at time t, it is linearly interpolated in time on the current mesh from two of these pre-calculated ALE metric fields. This method has two drawbacks: it requires computing and saving all these ALE metric fields and it requires a metric interpolation stage at each mesh optimization which leads to consequent memory, I/Os and CPU time overhead. The second possibility is to modify the input ALE metric field according to the current mesh deformation to get a consistant metric field at time t. To deform the continuous mesh defined at the beginning of the sub-interval, we adopt the same reasoning as the one that leads to the optimal instantaneous ALE continuous mesh in Section 1.3.2, but on a reverse time frame.

Let us consider t^k a time in the sub-interval $[t^i, t^{i+1}]$ and $\mathbf{d}$ the displacement field of the mesh between t^i and t^k. Here, the problem is reversed, we seek for the continuous mesh at time t^k that will result in the continuous mesh at time t^i once moved with displacement $-\mathbf{d}$. At time t^i, the continuous mesh is $\mathbf{M}_{L^p}^{i,\mathrm{ALE}}$ (the adapted mesh at time t^i is generated directly using this metric field).

Displacement $\mathbf{d}$ is the displacement of vertices between t^i and t^k: $\mathbf{x}^k = \mathbf{x}^i + \mathbf{d}(\mathbf{x}^i)$ and we write $\mathbf{d}'(\mathbf{x}^k) = \mathbf{x}^i - \mathbf{x}^k = -\mathbf{d}(\mathbf{x}^i)$. We define :

$$\begin{array}{rccl} \boldsymbol{\phi'}: & \Omega^k & \longrightarrow & \Omega^i \\ & \mathbf{x}^k & \longmapsto & \mathbf{x}^i = \boldsymbol{\phi'}\left(\mathbf{x}^k\right) = \mathbf{x}^k + \mathbf{d}'(\mathbf{x}^k)\,. \end{array} \tag{3.20}$$

We start from an adapted mesh which is unit for metric field $\mathbf{M}_{L^p}^{i,\mathrm{ALE}}$ at time t^i and we search for the expression of the metric field at time t^k for which the deformed adapted mesh at time t^k is unit. Following Section 1.3.2, we consider an edge $\mathbf{e}^k$ at time t^k having edge $\mathbf{e}^i$ as image by $\boldsymbol{\phi'}$ at time t^i. They verify:

$$\begin{aligned} 1 &= \left(\mathbf{e}^i(\mathbf{x}^i)\right)^T\ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}^i\right)\ \left(\mathbf{e}^i(\mathbf{x}^i)\right) \\ &= \left[\mathbf{e}^i\left(\boldsymbol{\phi'}(\mathbf{x}^k)\right)\right]^T\ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}^i\right)\ \mathbf{e}^i\left(\boldsymbol{\phi'}(\mathbf{x}^k)\right) \\ &= \left(\left[\nabla^k\boldsymbol{\phi'}(\mathbf{x}^k)\right]^T\ \mathbf{e}^k(\mathbf{x}^k)\right)^T\ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}^i)\ \left(\left[\nabla^k\boldsymbol{\phi'}(\mathbf{x}^k)\right]^T\ \mathbf{e}^k(\mathbf{x}^k)\right) \\ &= \left(\mathbf{e}^k(\mathbf{x}^k)\right)^T\ \left\{\nabla^k\boldsymbol{\phi'}(\mathbf{x}^k)\ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}(\mathbf{x}^i)\ \left[\nabla^k\boldsymbol{\phi'}(\mathbf{x}^k)\right]^T\right\}\ \left(\mathbf{e}^k(\mathbf{x}^k)\right)\,. \end{aligned}$$

In other words, the metric field $\mathbf{M}_{L^p}^{i,optim}$ we are looking for at time t^k is:

$$\mathcal{M}_{L^p}^{i,optim}(\mathbf{x}(t^k)) = \nabla^k\boldsymbol{\phi'}\left(\mathbf{x}(t^k)\right)\ \mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}(t^i)\right)\ \left[\nabla^k\boldsymbol{\phi'}\left(\mathbf{x}(t^k)\right)\right]^T\,. \tag{3.21}$$

This metric field is easy to compute on the fly, because the central term is the input metric field $\mathcal{M}_{L^p}^{i,\mathrm{ALE}}\left(\mathbf{x}(t^i)\right)$ which is known, and the gradients of $\boldsymbol{\phi'}$ are easy to compute, since the displacement considered in $\boldsymbol{\phi'}$ is the opposite of the current displacement at each vertex. Note that the gradients of $\boldsymbol{\phi'}$ should be computed on $\Omega(t^k)$ that is to say on the current mesh.

3.4.2 Modification of the algorithm

The overall global fixed-point mesh adaptation algorithm remains unchanged from [Barral 2015]. the simulation time frame is still divided into sub-intervals, a global fixed-point strategy is used to converge the meshes and the solutions within a sub-interval and the mesh is moved using the connectivity-change moving mesh algorithm presented On each sub-interval, the mesh number of vertices remains the same. One or several mesh deformation steps are performed during a sub-interval. The connectivity-change moving mesh algorithm uses mesh optimizations to make the mesh deformation efficient and robust. These mesh optimizations, smoothing and swapping operators, use the dynamic (corrected) metric field to be compliant with the current dynamic adapted mesh.

The difference concerns the computation of the ALE Hessian-metric for the next adaptation step. ALE Hessian-metrics $|H_u^{i,\mathrm{ALE}}|$ need to be sampled in time. To compute each sample,

we use the displacement given by the mesh deformation step without taking into account the smoothing optimization. In numerical experiments, we did not see any significant difference on the solution accuracy and the quality of the adapted meshes by considering or not the smoothing in the ALE Hessian-metrics computations. This is due mainly to the fact that the mesh smoothing has only a slight influence in 3D, the mesh displacement is predominantly governed by the mesh deformation [Alauzet 2014]. The mesh smoothing is only a slight perturbation around the mesh deformation displacement to get better shaped elements. Nevertheless, we prefer not to consider the smoothing displacement when computing ALE Hessian-metrics because:

- the geometry displacement is clearly defined, thus the mesh deformation is converging toward a unique mesh deformation and therefore we converge toward a unique optimal ALE continuous mesh

- the smoothing correction for one node may change from one fixed-point iteration to the other, thus for each node it does not converge to a fixed correction.

The gradient of the mesh deformation displacement field is computed on the mesh position at the beginning of the sub-interval. The Hessian of the sensor is computed on the mesh current position and is mapped back to the mesh position at the beginning of the sub-interval. Then, as in the standard algorithm, the ALE Hessian-metrics are computed at each vertex.

Note that the connectivity-change moving mesh algorithm allows the mesh vertices to move to their final position given by the mesh deformation solution while it would be impossible to reach such positions without skewing mesh elements with a classical method. Thanks to the robustness of this algorithm, we are always able to calculate the ALE hessian-metrics.

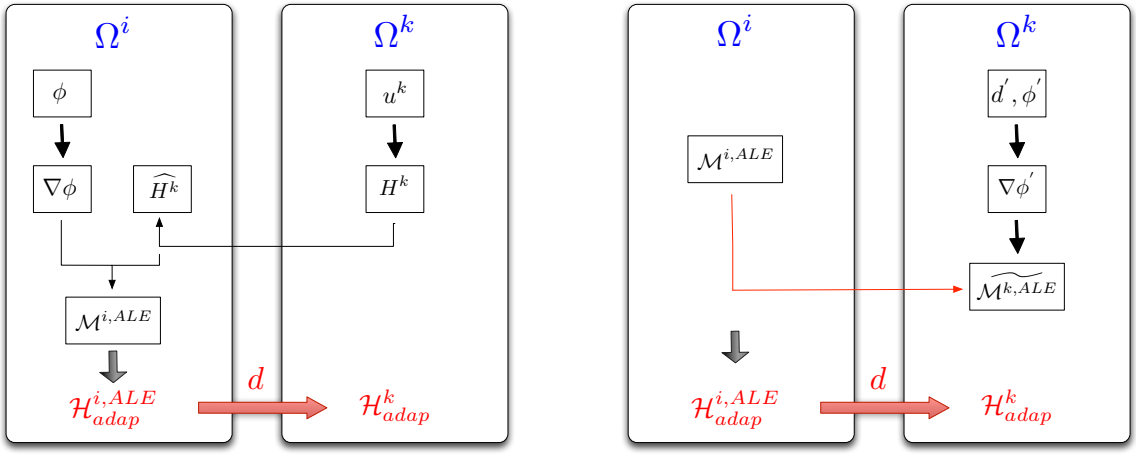


Figure 3.1: Fom left to right : Classical metric, ALE metric

3.5 Validation with analytic solutions of the metric for optimizations

The previous result gives us the optimal metric that will lead to an adapted mesh once moved with a given displacement. Let us illustrate this result on analytic examples. Analytic examples are only given in 2D.

3.5.1 Procedure

Let $\mathcal{H}_0^n$ be an uniform mesh, $\mathbf{d}$ a displacement field between two times t^n and t^{n+1}, and u^{n+1} a sensor at the final time. The goal is to generate a mesh $\mathcal{H}^n$ at the initial time that, once moved into $\mathcal{H}_{ALE}^{n+1} = \mathbf{d}(\mathcal{H}^n)$ will be adapted to the sensor. At the same time, a reference adapted mesh is computed, that is directly adapted to sensor u^{n+1}. To do so, the classical steady-state procedure described in Algorithm 8. Since the sensor is analytic and to be fair with the ALE version, only one iteration of the algorithm is performed. The metric used is normalized to look like the regular optimal metric for unsteady simulations, obtained from the hessian of the sensor. In what follows, we note $\mathcal{H}_{ALE}^{n+1}$ the mesh obtained with the ALE procedure and $\mathcal{H}_{ref}^{n+1}$ the mesh directly adapted.

$$
\begin{aligned}
\nabla^n \boldsymbol{\phi} &= \texttt{ComputeTransformationGradient}(\mathcal{H}_0^n, \mathbf{d}) \\
\mathcal{H}_0^{n+1} &= \texttt{MoveMesh}(\mathcal{H}_0^n, \mathbf{d}) \\
u^{n+1} &= \texttt{ComputeTargetSensor}(\mathcal{H}_0^{n+1}) \\
\mathcal{M}_{L^p}^{n,ALE} &= \texttt{ComputeALEMetric}(\mathcal{H}_0^{n+1}, u^{n+1}, \nabla^n \boldsymbol{\phi}) \\
\mathcal{H}^n &= \texttt{AdaptMesh}(\mathcal{H}^n, \mathcal{M}_{L^p}^{n,ALE}) \\
\mathcal{H}_{ALE}^{n+1} &= \texttt{MoveMesh}(\mathcal{H}^n, \mathbf{d})
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{M}_{L^p}^{n+1} &= \texttt{ComputeMetric}(\mathcal{H}_0^n, u^{n+1}, H_{u^{n+1}}) \\
\mathcal{H}_{ref}^{n+1} &= \texttt{AdaptMesh}(\mathcal{H}_0^n, \mathcal{M}_{L^p}^{n+1}) \,.
\end{aligned}
$$

According to the above developments, we expect the mesh $\mathcal{H}^{n+1}$, obtained by moving the vertices of $\mathcal{H}^n$ with $\mathbf{d}$, to be optimal for the control of the interpolation error of u^{n+1} in L^1 norm.

To study the adapted characteristic of the resulting meshes, two quantities are considered:

- the mesh quality, computed using the metric $\mathcal{M}_{L^p}^{n+1}$ of the reference adapted mesh (*i.e.* the one that was adapted directly, ... The quality measure is the one recalled in Section 1.1.2, Equation (1.19).

- the error comitted when the sensor function is projected on the mesh remains the best way to evaluate if the mesh is really adapted to a sensor. Here we consider the interpolation error $\|\Pi_h u - u\|$ for sensor u, where $\Pi_h u$ is the piecewise P^1 function with $\Pi_h u(P_i) = u(P_i)$ for any vertex P_i of the mesh.

The displacement considered are straight-line displacements of somewhat large amplitude, whereas infinitesimal displacements have been considered in the theoretical analysis. However, this is closer to real life simulations, with large displacements even within a sub-interval. Note that the ALE metric does not guarantee that the mesh moved at time t^{n+1} is still valid. In practice, mesh optimizations deal with this issue and ensure the validity of the mesh together with its quality. In this study, no mesh optimization is performed on the adapted mesh, or during the adaptation or moving process, to avoid altering the results of the quality analysis. For the same reason, unlike in the actual adaptation loop, no gradation is performed.

3.5.2 Analytic function considered

The function here is taken from [Olivier 2011a] in order to confirm and complete the analysis proposed in the present work. Its sensor presents specific features (smoothness, discontinuities, features of different scales...) that are as many difficulties for the adaptation process, so that handling them correctly shows the robustness of the approach. As for the displacements, they are set so that the displacement is null on the boundaries.

In 2D. The sensor u is represented on Fig. 3.2:

$$u^{n+1}(x,y) = \begin{cases} 0.01 \sin(50xy) & \text{if } |xy| \geq \dfrac{2\pi}{50} \\ \\ \sin(50xy) & \text{if } |xy| < \dfrac{2\pi}{50} \end{cases}$$

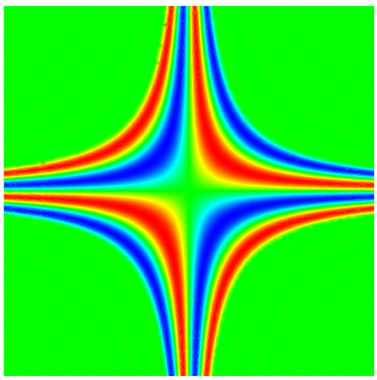


Figure 3.2: Initial sensor

and the displacement is represented on Fig. 3.3:

$$\mathbf{d}(x,y) = \begin{bmatrix} \begin{cases} -0.3\,(x+1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x \geq 0 \\ 0.3\,(x-1)\left(y^2-1\right)\exp\left(-5x^2\right), & \text{if } x < 0 \end{cases} \\ \\ \begin{cases} -0.3\left(x^2-1\right)(y+1)\exp\left(-5y^2\right), & \text{if } y \geq 0 \\ 0.3\left(x^2-1\right)(y-1)\exp\left(-5y^2\right), & \text{if } y < 0 \end{cases} \end{bmatrix}.$$

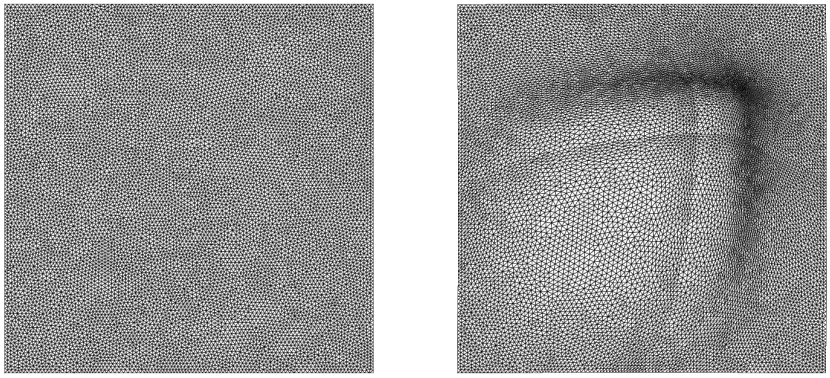


Figure 3.3: Initial Mesh is the uniform unstructured square mesh here is the representation of the displacement at t^{n+1}

Results

In Figure 3.4, the mesh on the left is the one that was adapted directly quality, computed using the metric $\mathcal{M}_{L^p}^{n+1}$ of the reference adapted mesh (*i.e.* the one that was adapted directly

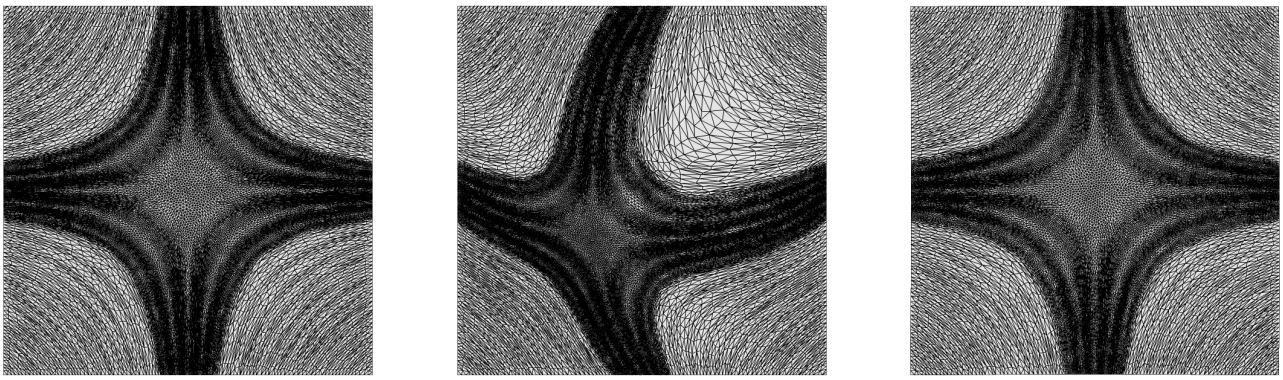


Figure 3.4: Fom left to right : Hessian metric at t^{n+1}, ALE Hessian metric at t^n, Moved and Updated ALE Hessian metric at t^{n+1}

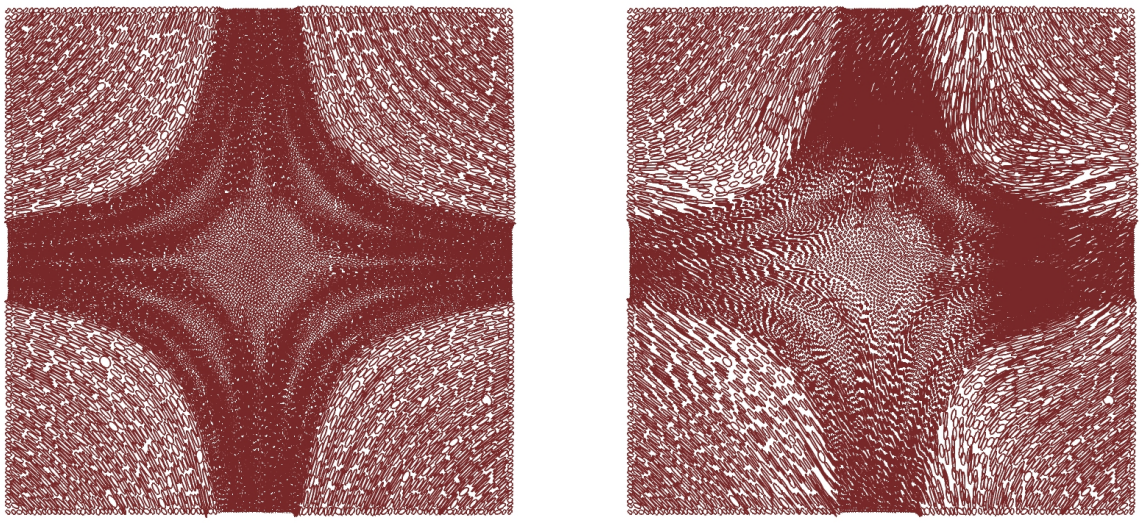


Figure 3.5: Fom left to right : Updated Metric , ALE Metric

3.5.3 Numerical 3D example : 3D F117 nosing up

This example is a subsonic notional F117 aircraft geometry nosing up, that creates a vortical wake. An inflow of air at Mach 0.4 arrives in front of the aircraft, initially in horizontal position, that noses up, stays up for a while, then noses down. In this example, the aircraft rotates around its center of gravity. Let $T = 1s$ be the characteristic time of the movement and $\theta^{max} = 20^o$ the

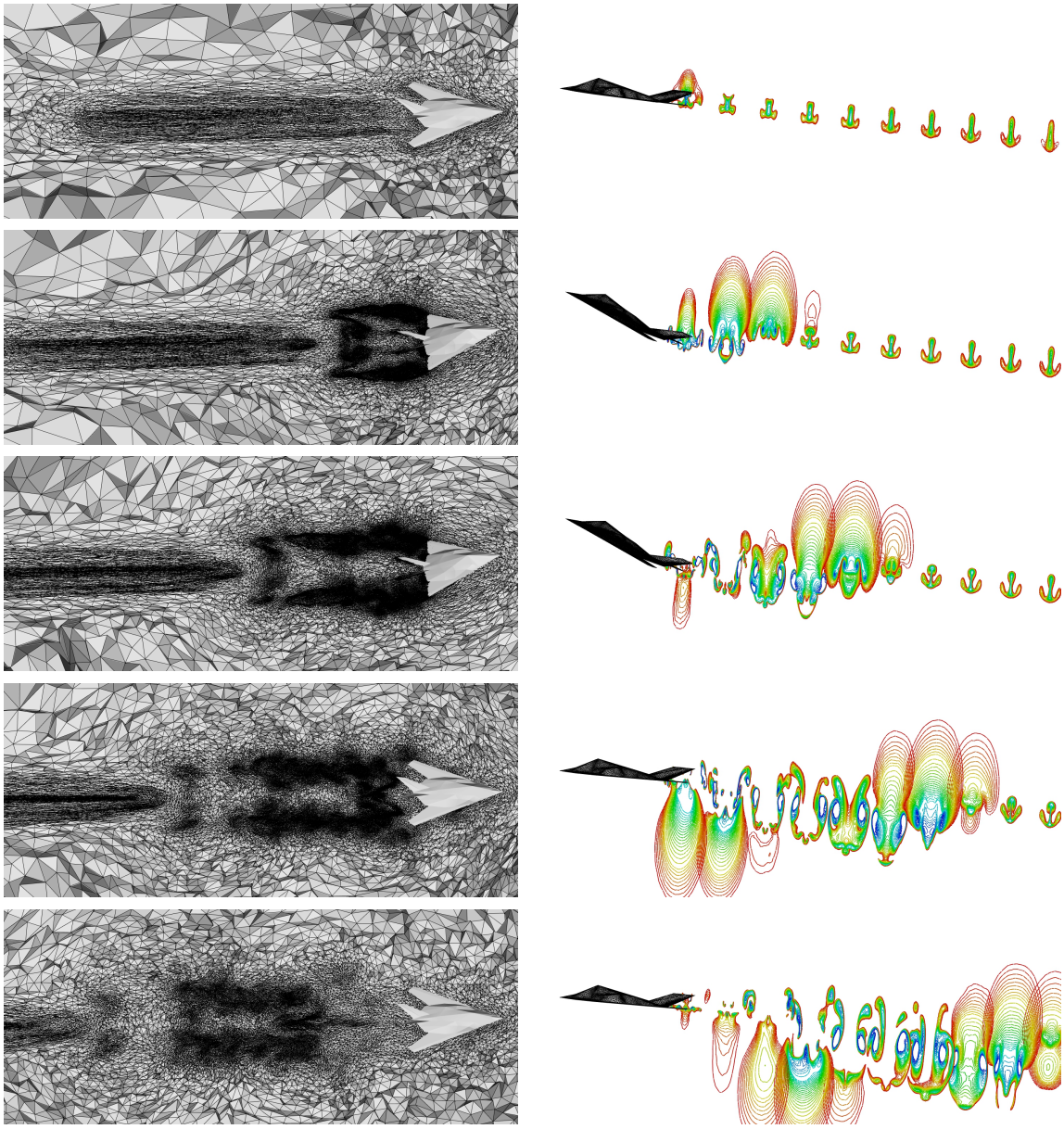


Figure 3.6: Nosing up F117 test case: Adapted meshes (view from the top) and mach isolines for the at different time steps.

maximal angle reached, the movement is defined by its angle of rotation, of which the evolution

is divided in 7 phases:

$$\theta(t) = \theta^{max} \begin{cases} 0 & \text{if} \quad 0 \le t \le T/2 \qquad (i) \\ \frac{2(t-\frac{T}{2})^2}{T^2} & \text{if} \quad \frac{T}{2} < t \le T \qquad (ii) \\ \frac{1}{2} + 2\,(\frac{2\,(t-\frac{T}{2})}{T} - 1) - \frac{1}{2}\,(\frac{4\,(t-\frac{T}{2})^2}{T^2} - 1) & \text{if} \quad T < t \le \frac{3T}{2} \qquad (iii) \\ 1 & \text{if} \quad \frac{3T}{2} < t \le \frac{7T}{2} \qquad (iv) \\ 1 - \frac{2\,(t-\frac{7T}{2})^2}{T^2} & \text{if} \quad \frac{7T}{2} < t \le 4T \qquad (v) \\ \frac{1}{2} - 2\,(\frac{2\,(t-\frac{7T}{2})}{T} - 1) + \frac{1}{2}\,(\frac{4\,(t-\frac{7T}{2})^2}{T^2} - 1) & \text{if} \quad 4T < t \le \frac{9T}{2} \qquad (vi) \\ 0 & \text{if} \quad \frac{9T}{2} t \le 5T \qquad (vii) \end{cases}$$

Phase (i) is an initialization phase, during which the flow around the aircraft is established. Phases (ii) and (iii) are respectively phases of accelerated and decelerated ascension. Vortices start to grow behind the aircraft, and they expand during phase (iv), where the aircraft stays in upward position. Phases (v) and (vi) are phases of accelerated and decelerated descent, the vortices start to move away and they slowly disappear in phase (vii). Free-stream conditions are imposed on the faces of the surrounding box, and slipping conditions on the aircraft.

The time simulation interval $[0, 5]$ was divided into 96 sub-intervals, and 5 fixed-point iterations are performed. The adaptation sensor is the local Mach number. A space-time complexity of 48 million was prescribed leading to a sub-interval average spatial complexity $\mathcal{N}_{avg} = 500,000$. The adapted meshes at the last fixed-point iteration have between $100,000$ and $1,400,000$ vertices, for an average number of vertices equal to $841,000$. A total of $n_{iter} = 38,980$ time-steps have been performed in the last fixed-point iteration. The total number of space-time vertices N_{st} at the last fixed-point iteration is equal to 35 billion. The total CPU time of this simulation is 95 hours. 31 hours being spent in the last fixed-point iteration (error estimate, metric gradation, adaptive mesh generation, solution interpolation, and flow solver).

Views of the adapted meshes and the corresponding solutions are displayed in Fig. 3.6. The vortical wake is propagated far from the aircraft, and the patterns of the vortices are highly resolved. The observation of the adapted meshes shows that they are actually refined only in the vicinity of the wake, and with such a precision that one can see the vortices being created, evolving and vanishing just by looking at the meshes. A view at several meshes within the same sub-interval shows that the mesh evolves continuously following the physical phenomena. In the wake far from the aircraft, the elements are highly anisotropic, whereas they have to be isotropic in the area of vorticity due to the characteristic of the physical phenomena.

3.6 Conclusion

This chapter addresses metric-based mesh adaptation for time dependent and dynamic meshes simulations. In the first sections, a brief survey of mesh adaptation is presented, and the metric-based adaptation concept is detailed, notably the error analysis and choose of metric optimization. This concept is used to derive an improved version of the previous unsteady fixed-point mesh adaptation algorithm. Notably, a global normalization term arises, which results in a global fixed-point algorithm. The method and recovery to compute the Hessian metric was considered. An efficient parallelization of the algorithm was carried out, which makes it possible to run a number of unsteady 3D simulations with an accuracy unthinkable a few years ago. The ALE metrics proposed in [Olivier 2011a] then in [Barral 2015] allow to generate meshes that will be adapted once moved with a certain displacement. Analytic examples validated this metric, and it was used within a new error analysis, to derive an ALE metric for adaptive sub-intervals. The regular time-dependent algorithm was modified to use this metric, and coupled with the moving mesh algorithm and example is done on a 2D analytic example.

The main contributions of this thesis with respect to pre-existing work in the GAMMA group is:

- An update strategy was proposed for the metric at mesh optimization times, to avoid spoiling the adaptation with optimizations.
- A study of the error on analytic case for the ALE metric that confirmed its adapted characteristic.
- A new version of the global fixed-point unsteady mesh adaptation algorithm was used.

Several perspective arise from this work, notably:

- To improve the adaptation algorithm, the next step is to control the temporal error. This would in particular lead to automatic adaptation of the size of the sub-intervals. This appears to be all the more important with moving meshes, where the size of the mesh is not even constant throughout a sub-interval. This improvement is necessary to deal with the ALE implicit scheme presented in 2.
- The convergence of this unsteady adaptative method needs to be studied, depending on the number of vertices and/or the size of the sub-intervals, as well as the influence of the initial solution on the final solution, and the ability of the algorithm to recover features of the solution lost due to coarse initial meshes.
- The metric used for optimizations during the adaptation loop should be analyzed in detail. Indeed, this step is crucial in the adaptation process, since it controls where the elements actually go.

- Vertices should be allowed to move on non plane surfaces. However, this is very complicated, since it requires reconstructing the surfaces with a good geometric accuracy to avoid introducing errors due to bad surface discretization.
- At longer term, adaptative moving mesh methods mentioned in [**?**] appear as an interesting alternative to prescribe the displacement of the vertices, and possible interactions with metric based unsteady adaptation should be investigated.

Chapter 4

Adjoint Methods

Adjoint techniques have gained widespread use over past decades as simulation capabilities have progressed. The ability to compute not only the simulation but to focus also on a functional output of interest has become increasingly recognized as a significant capability for enabling accuracy. We count nowadays many applications of the adjoint approach. The most classical ones are in control theory [Lions 1971], optimization [Allaire 2007], sensitivity analysis [Allaire 2015], inverse problems [A. Tikhonov 1977] and data assimilation [F.X. Le Dimet 1986]. The aim of this chapter is not to be exhaustive but to report on the universality nature of adjoint methods. Moreover understanding its use in other domains can lead to good ideas relevant to the implementation of our subject. In CFD, their utility is also reflected in numerous applications, mainly in design optimization [Giles 2000, Mohammadi 1997, Mavriplis 2000, A.Stück 2009] and less-commonly in mesh adaptation [Venditti 2002, Loseille 2010c, Belme 2012, Fidkowski 2017]. In the first domain, the adjoint method is the only way to compute a gradient and perform efficient optimization due to the large number of design variables. In the domain of mesh adaptation, the use of adjoint computation is not so common. It is therefore appropriate : if adjoint methods can be used to compute sensitivities of a functional then they can be used to determine regions of the computational domain that are most sensitive to a given functional and provide an error estimation in a specific quantity of interest, rather than an estimate of total simulation error. In this case, error estimation purposes, which in turned can be used to drive adaptive error control schemes in the interest of producing more accurate, reliable and predictive simulation outcomes. Among linearisation methods, adjoint methods demonstrate their efficiency. For many engineering problems, this is an important advantage as it allows to focus computational resources on optimizing the accuracy of the specific simulation objective while de-emphazing other aspects of the simulation which are less important for the objective of interest. The adjoint-based error method distinguishes from other approaches that may have little relation to errors in integral outputs of primary concern to the engineer. As an example, one can consider the wake behind an aircraft. In order to resolve the wake with accuracy, a finer grid is locally required. But it is often the case that the computed wake, located a chord or two downstream of the wing, passes into a region represented by a rather coarse grid. In this case, a mesh adaptation based on an usual error estimation, as seen on the previous chapter, would probably induce further refinement in this region. However, the effort of refinement

coud be useless if errors in this region has a very small contribution to computed lift and drag. Probably, refinement in region closer to the wing, near the leading and trailing edges would lead to much greater reduction in the lift and drag errors.

The use of adjoint methods for estimating errors in simulation output objectives is now fairly well-established, particularly for spatial discretization errors in steady-state aerodynamics problems [Venditti 2002, Loseille 2010c]. Additionally adjoint methods for unsteady problems have been pursued over the last decade by various contributors [Belme 2012, Fidkowski 2017]. But in spite of theses advances, adjoint methods have remained confined to relatively small subset of simulation problems and there is a need to extend these methods to more complex moving geometries problems to justify their interest.

4.1 State of the art of adjoint method

In this section, let's take at first a back step to consider the mathematical environment to know where the method comes from.

4.1.1 Mathematical point of view

In mathematics, the term "adjoint" applies in several situations and one can encounter this term in linear algebra, in functional analysis, but also in group theory or further in category theory. Their common particularity is that there is always a link to the principle of duality which establishes a full meaning of the term "adjoint". And duality is an old and complex principle because it enables to view a same problem with two different points of view. The interest of this rewrite is that the two problems are closely related and takes to the great advantage that for various reasons, it may sometimes be easier to solve the dual problem than the primal problem. But this is also which makes this own major difficulty : one have to also consider the primal problem with an other perspective : the dual problem and leads to two spaces, two equations, two sets of variables, etc.
Let's make an aside to illustrate the commodity but, in the same time, the complexity of duality with the linear duality in plane geometry. Anyone knows the assumption : *"Two different points can be joined by an unique line"*. The dual of this assumption is : *"Two different lines meet in one point"*. But the vision of classical Euclidian geometry does not allow to check this last assumption in every cases. In fact two parallel lines never join. The dual space associated to this dual assumption needs an addition of a point at infinity and is called the projective plane. Now the relationship between points and lines is perfectly symmetrical in this space.
It can be stated that in mathematical optimization theory, the adjoint method has its origin

in the theory of Lagrange multipliers. The first use was probably by the mathematician and one of the founder of the variational calculus J.L Lagrange. He introduced it for Lagrangian functions in dynamical systems. The method of Lagrange multipliers is a strategy for finding the local maxima or minima of a function under equality constraints. The great advantage of it is that the optimization method can be solved without explicit parametrization in terms of the constraints. We recall the method as follows.
Consider the problem : *Let's f and g have continuous first partial derivatives, maximize $f(\mathbf{x})$ under the condition $g(\mathbf{x}) = 0$.*
To solve this problem we introduce a new variable λ called a Lagrange multiplier and study the Lagrangian function defined by :

$$\mathcal{L}(\mathbf{x}, \lambda) = f(\mathbf{x}) - \lambda \cdot g(\mathbf{x}).$$

If $f(\mathbf{x_0})$ is a maximum of f for the constrained problem then there exists λ_0 such that $(\mathbf{x_0}, \lambda_0)$ is a stationary point for $\mathcal{L}$ (*i.e* its partial derivatives at this point are zero).

4.1.2 Interpretation

The Lagrangian multipliers measure the sensitivity of the optimal value by relation to the variation of the constraint. One way of looking at them is that they give the influence of an arbitrary source term on the functional of interest.

The adjoint space is chosen to simplify the physical interpretation of equation constraints. It may stands as a Hilbert space. In the field of functional analysis, each bounded linear operator on a Hilbert space has a corresponding adjoint operators.

4.1.3 Numerical adjoint approaches

The basic derivation for adjoint error estimation can be performed in several manners in the literature. And depending on the sphere of use and underlying motivations the adjoint could be considered as "continuous" or "discrete". Let's make in this section a quick overview of the use of adjoint in numerical computations before returning to the scope of output-based mesh adaptation.

Continuous adjoint approach : In the continuous formulation, the continuous governing equations are first derived using analytic differentiation and integration by parts and then linearized.
In functional analysis each bounded linear operator on a Hilbert space has a corresponding adjoint operator. Thus from functional analysis theory, a continuous system can be well-posed and considered in these terms and we could say that a continuous adjoint system can be derived

for any fuctional output. These however doesn't mean that all functionals lead to properly defined adjoint boundary conditions. Several works in litterature [Arian 1997, Castro 2007, A. Bueno-Orovio 2007, Anderson 1997] illustrate this problem and propose solutions usually by adding auxiliary boundary terms. In [Castro 2007], the authors conclude that for Euler flows, only output functional depending on pressure only can be considered. This choice remains a deep challenge when working with continuous adjoint formulation.

Discrete adjoint approach : In the discrete formulation, the governing primal equations are discretized then linearized and transposed for the adjoint problem. The discrete approach formulates directly the adjoint of the forward numerical scheme. This approach is very attractive since these dicrete adjoints can be in general generated automatically.
For discrete adjoint systems, the functionals area choice does not seem to be a problem and as far as the discrete adjoint system is well-posed, any admissible functional (in terms of non-null integral) can be considered.

For consistent discretizations, these two approaches converge to the same result while increasing the resolution of the discretization. However, the consistency of the discretization is not always guaranteed and needs to be performed. For some substantial works in the literature about this consistency for various schemes one can refer to [A. Sei 1995, Giles 2002].

Automatic Differentiation : Automatic Differentiation (AD) is a technique which automatically evaluates the derivatives of a function defined by a computer program. The AD exploits the fact that every computer program executes a sequence of elementary arithmetic operations and functions. To differentiate the code, it applies the chain rule repeatedly to these. Derivatives of arbitrary order can be then be computed automatically. For a code computing $\Psi_h(W_h)$ it can compute it as

$$\frac{\partial \Psi_h}{\partial W_h}\, U \quad \text{the forward mode (stands for forward accumulation)}$$

or

$$\left(\frac{\partial \Psi_h}{\partial W_h}\right)^T V \quad \text{the reverse mode (stands for reverse accumulation).}$$

Automatic Differentiation generally takes as input a computer source program, plus a request for differentiation and then returns the differentiated source program that evaluates the required derivatives. Several authors use Automatic Differentiation to generate adjoint CFD codes as [M. Martinelli 2010]. For numerical applications, Becker and Rannacher [R. Becker 2001] and Giles and Pierce [Giles 1999, N.A. Pierce 2004] showed the efficiency of adjoint systems for a better prediction of an output functional associated with a given system of PDE. Finally, let's cite some of AD tools : `ADIFOR` [Bischof 1998],

`TAPENADE` [Hascoet 2004], `ODYSSEE` [C. Faure 1998], `OpenAD/F` [J. Utke 2008] and more recently package for Python programs as `PyCppAD` or `PyADOL-C` [Walter].

Our approach : Although continuous or discrete adjoint approaches can be applied in the case of mesh adaptation, in this thesis and to match with the discrete formal solver, we focus on the discrete one for implementation. However, none of the Automatic Differentiation tools is used because the adjoint state has to be highly correlated to the mesh even more with a moving mesh. It is computed in the in-house code solver `Wolf`.

4.2 Computation of adjoint Euler system in ALE formulation

In this section we detail the discrete adjoint system associated to unsteady inviscid Euler flows computed with an ALE formulation [E. Gauci 2017].

4.2.1 Unsteady adjoint Euler equations in ALE formulation

We consider the 3D unsteady compressible Euler equations for a Newtonian fluid in their ALE formulation set in the space-time computational domain $\mathcal{Q} = \Omega(t) \times [0, T]$, where T is the (positive) maximal time. Assuming that the gas is perfect, inviscid and that there is no thermal diffusion, the continuous unsteady Euler formulation of the equations is written for ALE formulation :

$$\Psi(W) = W_t + \nabla \cdot \mathcal{F}_{ale}(W) = 0 \tag{4.1}$$

$$\begin{cases} W = (\rho, \rho\mathbf{u}, \rho E)^T \text{ is the conservative variables vector} \\ \mathcal{F}_{ale}(W) = \mathcal{F}(W) - W \otimes \boldsymbol{w} \quad \text{where } \boldsymbol{w} \text{ is the mesh velocity} \\ \mathcal{F}(W) = (\rho\mathbf{u},\ \rho u\mathbf{u} + pe_1,\ \rho v\mathbf{u} + pe_2,\ \rho w\mathbf{u} + pe_3,\ (\rho E + p)\mathbf{u}) \text{ is the flux tensor} \end{cases}$$

and we have noted ρ the density of the fluid, p the pressure, $\mathbf{u} = (u, v, w)$ its Eulerian velocity, $E = 1/2\, q^2 + \varepsilon$ the total energy per unit mass, $q = \|\mathbf{u}\|$ and ε the internal energy per unit mass.

4.2.2 Discrete adjoint state in ALE formulation

Although any consistent approximation of the continuous adjoint system could be built by discretizing the continuous adjoint equation, we choose the option to build the discrete adjoint system from the discrete state system in order to be numerically closer to the true error. In this thesis we use the same notations as in [Belme 2012].

Explicit time integration scheme

Consider the following semi-discrete unsteady compressible Euler model (explicit RK1 time integration):

$$\Psi_h^n(W_h^n, W_h^{n-1}) = \frac{W_h^n - W_h^{n-1}}{\delta t^n} + \Phi_{ale,h}(W_h^{n-1}) = 0 \quad \text{for } n = 1, ..., N. \tag{4.2}$$

The time-dependent functional is discretized as follows. The functional doesn't depend on the mesh displacement :

$$j_h(W_h) = \sum_{n=1}^{N} \delta t^n j_h^{n-1}(W_h^{n-1}).$$

For the sake of simplicity, we restrict to the case $g_T = 0$ for the functional output defined. The problem of minimizing the error committed on the target functional $j(W_h) = (g, W_h)$, subject to the Euler system (4.2), can be transformed into an unconstrained problem for the following Lagrangian functional [Giles 2000]:

$$\mathcal{L}(W_h, W_h^*) = \sum_{n=1}^{N} \delta t^n j_h^{n-1}(W_h^{n-1}) - \sum_{n=1}^{N} \delta t^n (W_h^{*,n})^T \Psi_h^n(W_h^n, W_h^{n-1}),$$

where $W_h^{*,n}$ are the N vectors of the Lagrange multipliers (which are the time-dependent adjoint states). The conditions for an extremum are:

$$\frac{\partial \mathcal{L}}{\partial W_h^{*,n}} = 0 \quad \text{and} \quad \frac{\partial \mathcal{L}}{\partial W_h^n} = 0, \quad \text{for } n = 1, ..., N.$$

The first condition is clearly verified from Relation (4.2). Thus the Lagrangian multipliers $W_h^{*,n}$ must be chosen such that the second condition of extrema is verified. This provides the unsteady discrete adjoint system:

$$\left\{ \begin{array}{rcl} W_h^{*,N} & = & 0 \\ W_h^{*,n-1} & = & W_h^{*,n} + \delta t^n \dfrac{\partial j_h^{n-1}}{\partial W_h^{n-1}}(W_h^{n-1}) - \delta t^n (W_h^{*,n})^T \dfrac{\partial \Phi_{ale,h}}{\partial W_h^{n-1}}(W_h^{n-1}), \end{array} \right. \tag{4.3}$$

or equivalently, the semi-discrete unsteady adjoint model reads:

$$\Psi_h^{*,n}(W_h^{*,n}, W_h^{*,n-1}, W_h^{n-1}) = \frac{W_h^{*,n-1} - W_h^{*,n}}{-\delta t^n} + \Phi_{ale,h}^*(W_h^{*,n}, W_h^{n-1}) = 0 \quad \text{for } n = 1, ..., N$$

with

$$\Phi_{ale,h}^*(W_h^{*,n}, W_h^{n-1}) = \frac{\partial j_h^{n-1}}{\partial W_h^{n-1}}(W_h^{n-1}) - (W_h^{*,n})^T \frac{\partial \Phi_{ale,h}}{\partial W_h^{n-1}}(W_h^{n-1}).$$

4.2.3 Implicit scheme

We present in this paragraph the implicit scheme for unsteady adjoint Euler equation. Only the discrete adjoint solution which satisfies the implicit equation system is presented. The numerical implementation has not been done in the solver.

Consider the following unsteady Euler model using implicit time integration :

$$\Psi^n(W^{n+1}, W^n) = \frac{W^{n+1} - W^n}{\delta t^n} + \Phi(W^{n+1}) = 0, \text{ where } \Phi \text{ is the Euler flux functions,}$$

and a time-dependent functional :

$$j(W) = \sum_{n=1}^{N} \delta t^n j^n(W^n)$$

The problem of minimizing the error committed on the target functional $J^n(W^n)$, subject to the Euler system, can be solved using the method of Lagrange multipliers.

In this sense, we introduce the adjoint state W^* which stands as a Lagrange multiplier and the Lagrangian defined by :

$$\mathcal{L}(W, W^*) = \sum_{n=1}^{N} \delta t^n j^n(W^n) - \sum_{n=1}^{N} \delta t^n (W^{*,n})^T \ \Psi^n(W^{n+1}, W^n),$$

and study :

$$\frac{\partial \mathcal{L}}{\partial W^{*,n}}(W, W^*) = 0 \qquad \text{and} \qquad \frac{\partial \mathcal{L}}{\partial W^n}(W, W^*) = 0.$$

The first condition is clearly verified. The adjoint state is thus chosen such that the second condition is verified.

$$\begin{aligned}
&\frac{\partial \mathcal{L}}{\partial W^n}(W, W^*) \\
&= \delta t^n \frac{\partial j^n}{\partial W^n}(W^n) - \delta t^n (W^{*,n})^T \frac{\partial \Psi^n}{\partial W^n}(W^n, W^{n+1}) - \delta t^{n-1} (W^{*,n-1})^T \frac{\partial \Psi^{n-1}}{\partial W^n}(W^{n-1}, W^n) \\
&= \delta t^n \frac{\partial j^n}{\partial W^n}(W^n) + W^{*,n} - W^{*,n-1} - \delta t^{n-1} (W^{*,n-1})^T \frac{\partial \Phi}{\partial W^n}(W^n) \\
&= 0.
\end{aligned}$$

This equation implies :

$$\frac{W^{*,n-1} - W^{*,n}}{\delta t^n}$$

$$
\begin{aligned}
&= \frac{\partial j^n}{\partial W^n}(W^n) - \frac{\delta t^{n-1}}{\delta t^n}(W^{*,n-1})^T \frac{\partial \Phi}{\partial W^n}(W^n) \\
&= \frac{\partial j^n}{\partial W^n}(W^n) - \frac{\delta t^{n-1}}{\delta t^n}\left(W^{*,n-1} - W^{*,n}\right)^T \frac{\partial \Phi}{\partial W^n}(W^n) - \frac{\delta t^{n-1}}{\delta t^n}(W^{*,n})^T \frac{\partial \Phi}{\partial W^n}(W^n).
\end{aligned}
$$

Let us divide by $\frac{\delta t^{n-1}}{\delta t^n}$, we get :

$$
\left[\frac{Id}{\delta t^{n-1}} + \frac{\partial \Phi}{\partial W^n}(W^n)\right]^T \left(W^{*,n-1} - W^{*,n}\right) = \frac{\delta t^n}{\delta t^{n-1}} \frac{\partial j^n}{\partial W^n}(W^n) - (W^{*,n})^T \frac{\partial \Phi}{\partial W^n}(W^n).
$$

This provides the unsteady discrete adjoint model :

$$
\Psi^{*,n}(W^{*,n}, W^{*,n-1}, W^n) = \left[\frac{Id}{-\delta t^{n-1}} - \frac{\partial \Phi}{\partial W^n}(W^n)\right]^T \left(W^{*,n-1} - W^{*,n}\right) + \Phi^*(W^{*,n}, W^n) = 0,
$$

with

$$
\Phi^*(W^{*,n}, W^n) = \frac{\delta t^n}{\delta t^{n-1}} \frac{\partial j^n}{\partial W^n}(W^n) - (W^{*,n})^T \frac{\partial \Phi}{\partial W^n}(W^n).
$$

In the following of this thesis, we only consider the explicit time integration scheme.

4.3 Connectivity-change moving mesh strategy

The overall connectivity-change moving mesh algorithm in the primal state was described in chapter 2 section 2.1 in Algorithm 5. When coupled with a flow solver (see Sections 2.2), the flow solver is called after the optimization phase. To compute the adjoint state and to be consistent with the connectivity-change moving mesh algorithm in the primal state, a choice must be done for the backward connectivity-change moving mesh algorithm. THe choice made was :

- **Store all the coordinates of the vertices** at each iteration during the forward loop in order to not recompute the deformation during the backward loop
- **Not store the connectivities** but remake connectivity-change optimizations during the backward loop.

The first point is very costly in term of memory as we already have to store the states $\left\{W_h^{n-1}\right\}_{n=1,N}$ to solve the adjoint states. But in this way, the vertices make exactly the reverse movement. However, our choice has been to not store all the connectivities even if it would have been possible by storing a `char` by edge. Thus during the backward loop optimizations have remade over time. For this reason, we can have different connectivities between forward and backward meshes at the same corresponding time. Let's detail the process.

During the computation of the equation state, the forward meshes are deformed. Each iteration a connectivity-change occurs a tag $d^{*,opt}$ is used. Therefore, during the computation of the adjoint equation state, the backward meshes are moved. The initial mesh in backward computation is the last mesh of the forward computation. It is moved with the opposite displacement and if an iteration is tagged, an optimization of the mesh is performed. The backward meshes are then topologically equivalent to the forward meshes but not identical.

An example of forward mesh and backward mesh at the same time iteration is given in Figure 4.1. The case test used is the FSI Case 2D shock-ball interaction in a tube detailed in Section 4.8. The positions of the vertices are the same but the connectivity is different.

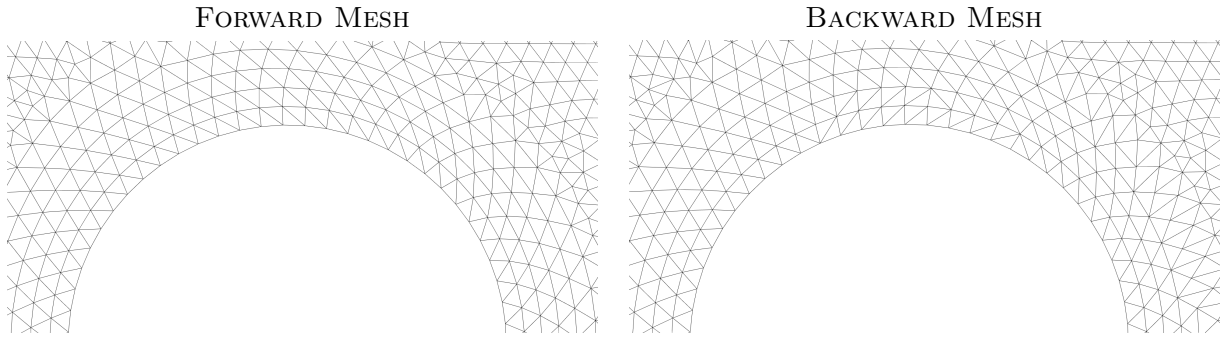


Figure 4.1: Exemple of forward mesh and backward mesh at adimentionned time $t = 130$.

4.4 The Discrete Geometric Conservation Law

As said in Section 2.2.3 for the forward mesh computation, we need to ensure that the movement of the mesh backward is not responsible for any artificial alteration of the adjoint computation, or at least, to make our best from a numerical point of view for the mesh movement to introduce an error of the same order as the one introduced by the numerical scheme.

The continuous adjoint state W^*, integrated on any arbitrary closed volume $C = C(t)$, writes :

$$- \int_{C(t)} W_t^* - \left(\frac{\partial \mathcal{F}_{ale}}{\partial W}\right)^* \nabla W^* \, \mathrm{d}\boldsymbol{x} - \int_{\partial C(t)} \left(\frac{\partial \mathcal{F}_{ale}}{\partial W}\right)^* W^* . \mathbf{n} \, \mathrm{d}\boldsymbol{s} \; = \; 0 \tag{4.4}$$

If Equation (4.4) is written for a constant adjoint state, we obtain :

$$- \frac{\mathrm{d}\left(|C(t)|\right)}{\mathrm{d}t} - \int_{\partial C(t)} (\boldsymbol{w} \cdot \mathbf{n}) \, \mathrm{d}\boldsymbol{s} = 0 \, . \tag{4.5}$$

As the constant adjoint state is a solution of the adjoint Euler equations, if boundaries transmit the flux towards the backward outside as it comes, we find a purely geometrical relation inherent to the continuous problem. For any arbitrary closed volume $C = C(t)$ of boundary $\partial C(t)$, Relation (4.5) is integrated into:

$$-\,|C(t+\delta t)| \,+\, |C(t)| = \int_t^{t+\delta t} \int_{\partial C(t)} (\boldsymbol{w}\cdot\mathbf{n})\,\mathrm{d}\boldsymbol{s}\mathrm{d}t, \quad \text{with } t \text{ and } t+\delta t \,\in\, [0,T]\,, \tag{4.6}$$

which is a reverse time of the usual Geometric Conservation Law (GCL). From a geometric point of view, this relation states that the algebraic variation of the volume of C between two instants in backward equals the algebraic volume swept by its boundary.

4.5 Backward Implementation

The properties of the backward implementation moving mesh has been detailed in the previous section. In the adjoint solver point of view, it is then necessary to introduce mesh velocities in the ALE formulation. In this section we detail the different Riemann solvers.

The implementation of the adjoint ALE solver was done in the in-house code `Wolf`.

4.5.1 Numerical flux computation

In the formula 4.3, the functional output is not affected by the ALE formulation. Its computation does not change from [Belme 2012]. The great difference focuses on the Jacobian solver computation. Let's detail some of them.

Jacobian in ALE Rusanov Riemann solver

$$\frac{\partial \Phi_{ale}^{Rusanov}(W_i, W_j, \mathbf{n_{ij}}, \sigma_{\mathbf{ij}})}{\partial W_j} = \frac{1}{2}\left(\frac{\partial F(W_i)\cdot \mathbf{n_{ij}}}{\partial W_j} - \sigma_{ij} Id - |\lambda_{ij}| Id\right). \tag{4.7}$$

where

$$|\lambda_{ij}| = \max(|\mathbf{W_i}\cdot\mathbf{n_{ij}}| + \mathbf{c_i} - \sigma_{\mathbf{ij}}, |\mathbf{W_j}\cdot\mathbf{n_{ij}}| + \mathbf{c_j} - \sigma_{\mathbf{ij}}).$$

Jacobian in ALE Roe Riemann solver

$$\frac{\partial \Phi_{ale}^{Roe}(W_i, W_j, \mathbf{n_{ij}}, \sigma_{\mathbf{ij}})}{\partial W_j} = \frac{1}{2}\left(\frac{\partial (F(W_j)\cdot \mathbf{n_{ij}})}{\partial W_j} - \sigma_{ij} Id - |\tilde{A}(W_i, W_j) - \sigma_{ij} Id|\right). \tag{4.8}$$

Jacobian in ALE HLLC Riemann solver

The Figure 4.2 illustrates the definition of our version of HLLC flux presented in section 2.2.2.

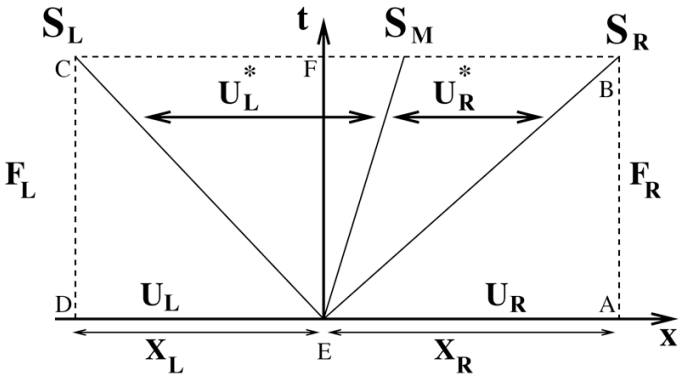


Figure 4.2: Region of integration containing the Riemann fan in Godunov's method

The HLLC flux depends on the waves speed, so does the associated Jacobian.

if $S_L > \sigma_{lr}$

$$\frac{\partial \Phi_{ale}^{HLLC}(W_l)}{\partial W_l} = \frac{\partial F(W_l)}{\partial W_l} - \sigma_{lr} Id \qquad \text{and} \qquad \frac{\partial \Phi_{ale}^{HLLC}(W_l)}{\partial W_r} = 0$$

if $S_L \leq \sigma_{lr} \leq S_M$

$$\frac{\partial \Phi_{ale}^{HLLC}(W_l^*)}{\partial W_l} = \frac{\partial F(W_l^*)}{\partial W_l} - \sigma_{lr} Id$$

and

$$\frac{\partial \Phi_{ale}^{HLLC}(W_l^*)}{\partial W_r} = \frac{\partial F(W_l^*)}{\partial W_r} - \sigma_{lr} Id$$

if $\sigma_{lr} > S_R$

$$\frac{\partial \Phi_{ale}^{HLLC}(W_r)}{\partial W_r} = \frac{\partial F(W_r)}{\partial W_r} - \sigma_{lr} Id \qquad \text{and} \qquad \frac{\partial \Phi_{ale}^{HLLC}(W_r)}{\partial W_l} = 0$$

Symmetrically,

if $S_L \leq \sigma_{lr} \leq S_M$

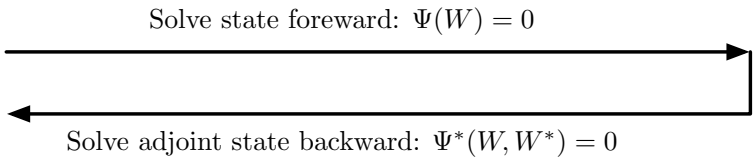
$$\frac{\partial \Phi_{ale}^{HLLC}(W_r^*)}{\partial W_r} = \frac{\partial F(W_r^*)}{\partial W_r} - \sigma_{lr} Id$$

and

$$\frac{\partial \Phi_{ale}^{HLLC}(W_r^*)}{\partial W_l} = \frac{\partial F(W_r^*)}{\partial W_l} - \sigma_{lr} Id$$

4.5.2 Memory management

As the adjoint system runs in reverse time, the first expression in the adjoint System (4.3) is referred to as adjoint "initialization".



Computing $W_h^{*,n-1}$ at time t^{n-1} requires the knowledge of state W_h^{n-1} and adjoint state $W_h^{*,n}$. Therefore, the knowledge of all states $\left\{W_h^{n-1}\right\}_{n=1,N}$ is needed to compute backward the adjoint state from time T to 0 which involves large memory storage effort. For instance, if we consider a 3D simulation with a mesh composed of one million vertices then we need to store at each iteration five millions solution data (we have 5 conservative variables). If we perform 1000 iterations, then the memory effort to store all states is 37.25 Gb for double-type data storage (or 18.62 for float-type data storage).

4.6 Dealing with memory

4.6.1 Packing and interpolation

After each swap step, a linear interpolation is performed to recover the solution. In other words, the solution at the vertices does not change, which is not conservative. The data of the finite volume cells (volume and interface normals) are then updated, together with the connectivity of the mesh (edges and elements). This approach however is DGCL compliant, since the constant state is preserved (in fact, any linear state is preserved). A better approach would be to use conservative interpolation [Alauzet 2010c] which is conservative and DGCL compliant and the best would be to develop the ALE formulation of the generalized swap in 3D from the formulation in 2D [Olivier 2011a].

4.6.2 Parallelization

Most parts of the code are parallelized for shared memory multi-core computers with a p-threads approach using a semi-automatic p-thread parallelization library [Maréchal 2016] coupled with a space filling curve renumbering strategy [Alauzet 2009].

The automatic parallelization library cuts the data into small chunks that are compact in terms of memory (because of the renumbering), then uses a dynamic scheduler to allocate the chunks to the threads to limit concurrent memory accesses. In the case of a loop performing the same operation on each entry of a table, the loop is split into many sub-loops. Each sub-loop

will perform the same operation (symmetric parallelism) on equally-sized portions of the main table and will be concurrently executed. It is the scheduler's job to make sure that two threads do not write on the same memory location simultaneously. When indirect memory accesses occur in loops, memory write conflicts can still arise. To deal with this issue, an asynchronous parallelization is considered rather than of a classic gather/scatter technique, *i.e.*, each thread writes in its own working array and then the data are merged.

However, with this approach, a subtle management of cache misses and cache-line overwrites is required to avoid drops in scaling factors. Space filling curves exhibit clustering properties very helpful for renumbering algorithms [Sagan 1994]. The Hilbert space filling curve based renumbering strategy is used to map mesh geometric entities, such as vertices, edges, triangles and tetrahedra, into a one dimensional interval. In numerical applications, it can be viewed as a mapping from the computational domain onto the memory of a computer. The local property of the Hilbert space filling curve implies that entities which are neighbors on the memory 1D interval are also neighbors in the computational domain. Therefore, such a renumbering strategy has a significant impact on the efficiency of a code. We can state the following benefits: it reduces the number of cache misses during indirect addressing loops, and it reduces cache-line overwrites during indirect addressing loops, which is fundamental for shared memory parallelism.

4.7 Validation of the adjoint code

Once the numerical adjoint scheme coded, the validation made is to check if the Jacobian fluxes in ALE formulation are equal to the variation rate of the fluxes in ALE formulation. For that we perform a validation test. We compute on one hand :

$$\Phi_{ale}(a + da) - \Phi_{ale}(a),$$

and on the other hand :

$$\frac{\partial \Phi_{ale}}{\partial W}(a) \cdot da.$$

The validation for 7 iterations of adjoint of the case test used is the FSI Case 2D shock-ball interaction in a tube is illustrated in Figure 4.3.

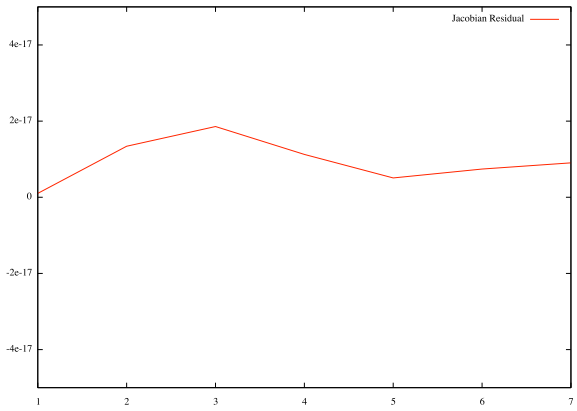


Figure 4.3: Residual $|\Phi_{ale}(a+da) - \Phi_{ale}(a) - \frac{\partial \Phi_{ale}}{\partial W}(a) \cdot da|$.

4.8 Numerical examples

4.8.1 Case with imposed movement : Naca0012 Rotation

This example was already presented in Chapter 2. We study the movement of a wing profile NACA0012 induced by an imposed of rotational displacement.

Mesh

In a circular domain of radius 20 is put a NACA0012 of length 1. The entire mesh includes 3291 vertices.

Initial conditions

To generate similar conditions of take-off, the initial Mach number chosen is 0.4.

Moving mesh

The moving of the NACA0012 is imposed. In phase 1, the coordinates of the geometry do not moved. The flow around the aircraft is established. Let $t_1 = 0.1$ be the time of the beginning of the movement and $\theta^{max} = 20^o$ the maximal angle reached at first time $t_{\theta^{max}}$. The movement of the wing profile is defined in phase 2 by calculating at each time step its angle of rotation $\alpha = 0.5 * \theta^{max} * (1 - \cos(\pi * (t - t_1)/(t_{\theta^{max}} - t_0)))$. The coordinates of the new points are then $x_{new} = x_{old} * \cos\alpha + y_{old} * \sin\alpha$ and $y_{new} = -x_{old} * \sin\alpha + y_{old} * \cos\alpha$. A phase 3 follows where the NACA0012 stays in upward position before the descent computed in the same way as phase 2.

Cost function

The considered functional output here is the drag of the surface S of the aircraft. The local

drag coefficient C_d leads to the global drag coefficient for a given body :

$$\int_S \frac{1}{S_{ref}} C_d(\mathbf{x}, t) \mathrm{d}\mathbf{x}$$

Numerical Results

Views of the two states : the primal state and the dual state for the considered functional output are displayed in Figure 4.4.

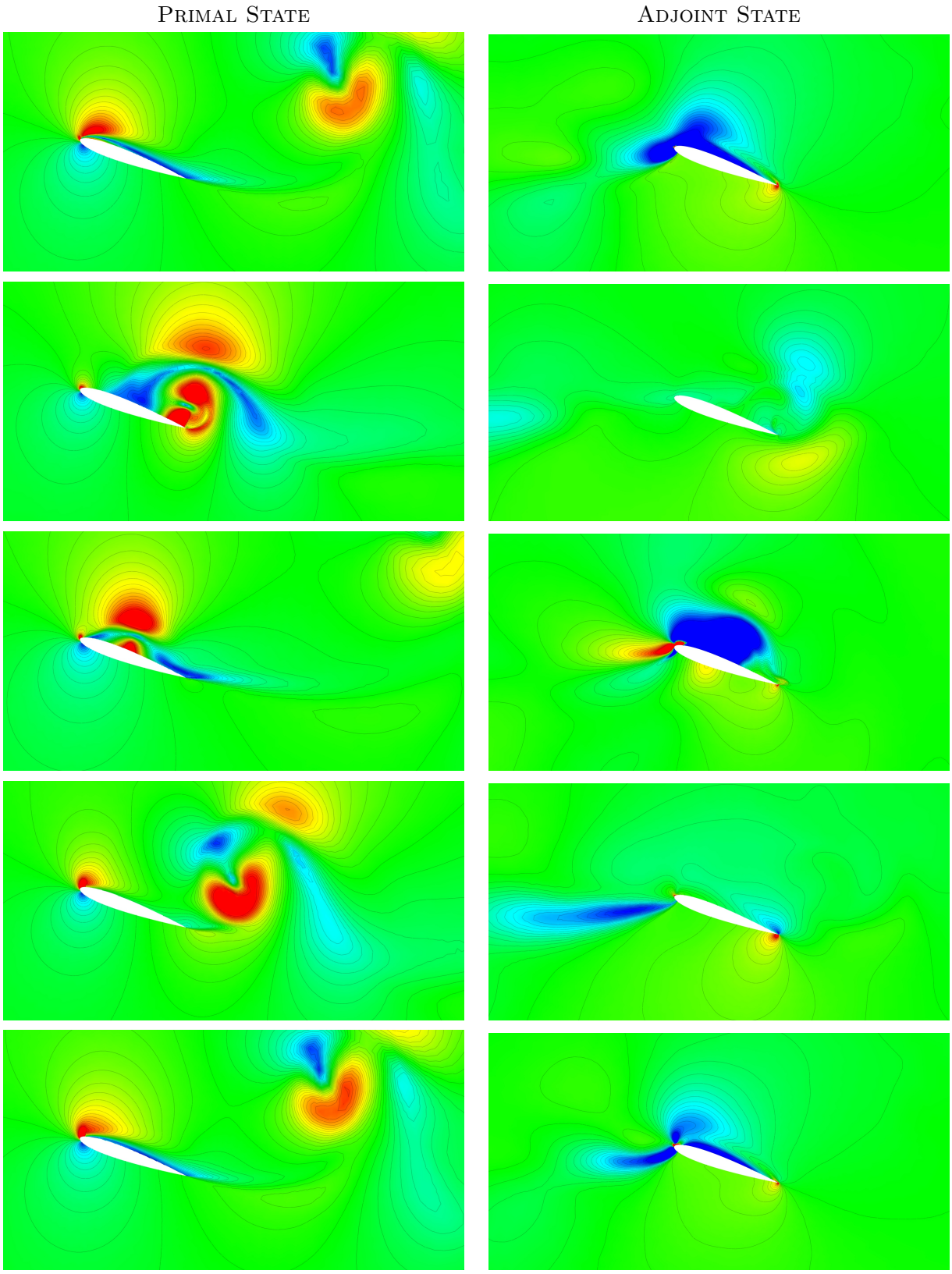


Figure 4.4: Nosing up NACA0012 test case: (view from the top to the bottom) Pressure at different time steps (t = 165, 135, 120, 105, 90, 75, 60, 45).

4.8.2 FSI Case : 2D Shock-ball interaction in a tube

This example is a variation of the classic Sod shock-tube problem. We consider a ball in a shock tube.

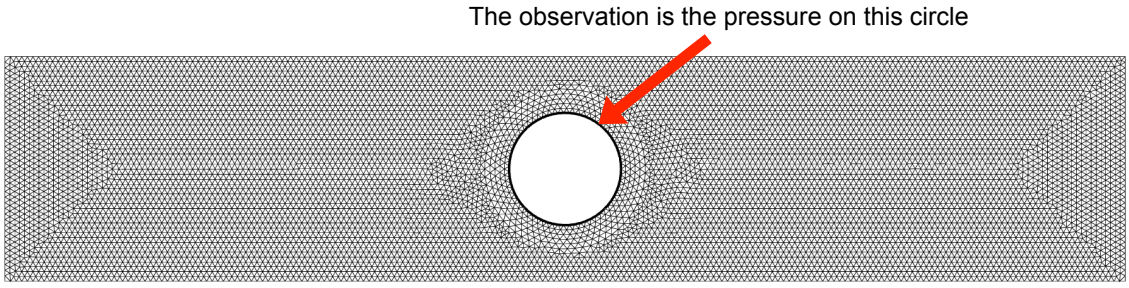
Mesh and initial conditions

The dimensions of the tube are: $[0,1] \times [0,0.2]$. At initial time, the gas is split in two states: for $x \leq 0.08$ the state is $(\rho_{left}, \mathbf{u}_{left}, p_{left}) = (1, \mathbf{0}, 4)$ whereas for $x > 0.08$ the state is $(\rho_{right}, \mathbf{u}_{right}, p_{right}) = (0, \mathbf{0}, 2.5)$. A ball of radius $r = 0.05$ is immersed in the gas, its center being in position $(0.5, 0.1)$. The tube is fixed. The simulation is run until final time $t_f = 0.5$. This initial state can be produced by having a diaphragm in the middle of the tube. The gas to the left and right of the diaphragm is initially at rest. The pressure and density are discontinuous across the diaphragm.

Cost function

The cost fonction considered is the pressure on the surface S of the ball. The output functional of interest is the quadratic deviation from ambient pressure on S :

$$j(W) = \int_0^T \int_S \frac{1}{2}(p(t) - p_{air})^2 \mathrm{d}S\mathrm{d}t.$$



Numerical results

At $t = 0$, the diaphragm is broken. The gas is then left to evolve freely and the classic rarefaction wave, contact discontinuity and shock wave appear. After a while, the ball is touched by the shock and the contact discontinuity, creating complex patterns both in front of it and in its wake.

The primal state and corresponding adjoint state at different time steps are shown in Figure 4.5.

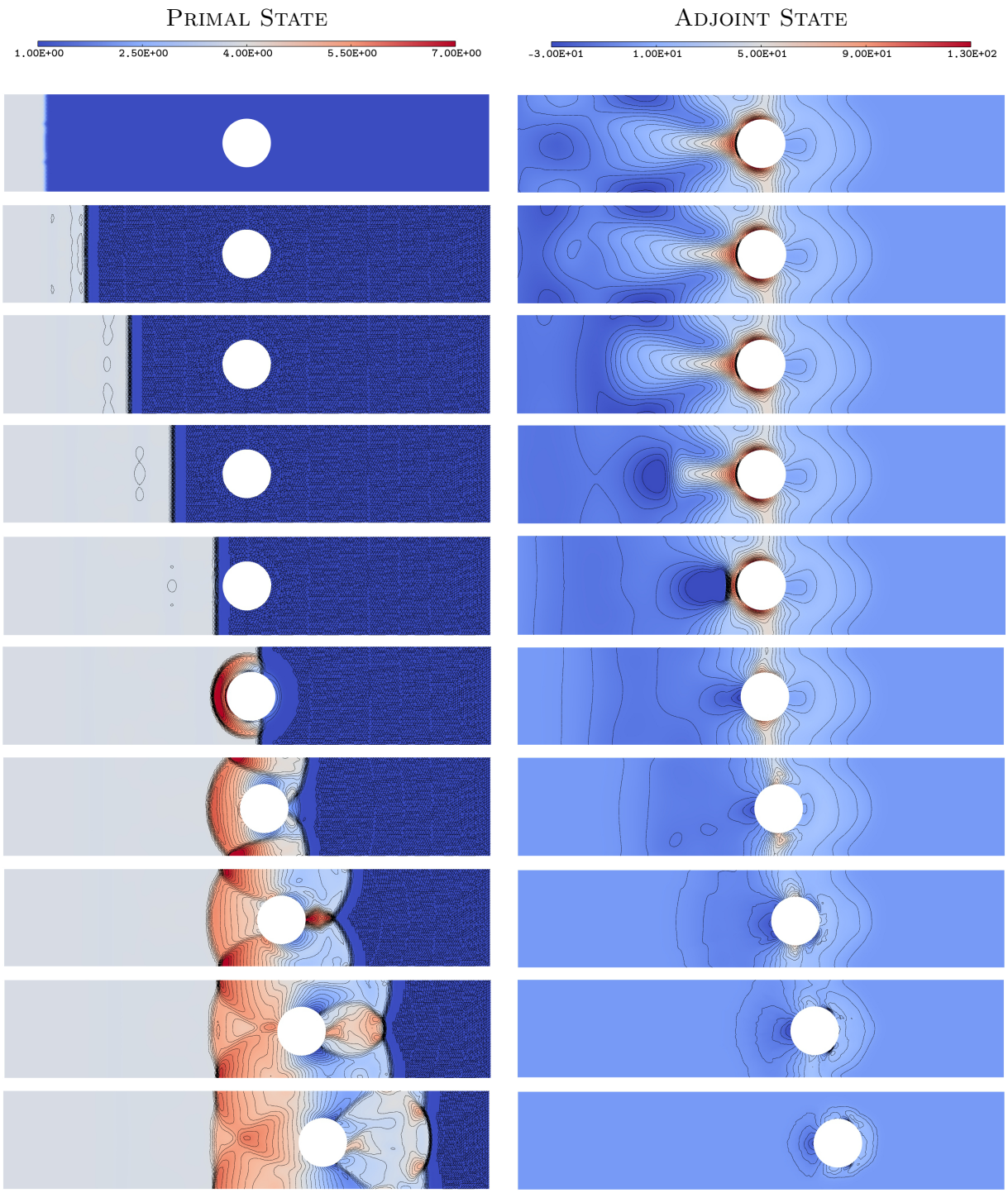


Figure 4.5: Ball in a shock tube test case: (view from the top to the bottom) Pressure at different time steps (t = 0, 0.025, 0.05, 0.075, 0.1, 0.125, 0.150, 0.175, 0.2, 0.225) and corresponding adjoint state.

4.8.3 Numerical qualitative verification in 3D : Lohner blast-like on a Bump

Meshes and initial conditions

This 2D case is a Löhner semi-circular blast-like simulation in a domain $[0,5] \times [0,0.5]$ with a sinus-like bump of diameter 0.45 and center 1.775. The equation of this bump is similar to the Nasa 2D Bump-in-channel. The bumps extends from 1.55 to 2. Its maximum height is $h = 0.2$. Its entire definition is :

$$y = \begin{cases} h * \sin^4\left(\pi(x - 1.55)/0.45\right), & \text{if } 1.55 < x < 2 \\ 0\,, & \text{else} \end{cases} .$$

It is then extruded into a 3D mesh of 261918 vertices and dimensions $[0,5] \times [0,0.5] \times [0,0.8]$. The bump is extruded in a bump-like cylinder.

Initial conditions

The half-circular blast is located on $(1.2, 0)$. In 3D, it is a half-cylinder blast of symmetrical axis $(1.2, 0, z)$. At initial time, the state is $(\rho_{blast}, \mathbf{u}_{blast}, p_{blast}) = (10, \mathbf{0}, 25)$ whereas for the rest of the domain the state is $(\rho_{domain}, \mathbf{u}_{domain}, p_{domain}) = (1, \mathbf{0}, 2.5)$.

Moving mesh

The moving mesh function is :

$$A = 0.15 \text{ is the amplitude of the oscillation}$$
$$\omega = \pi/0.7 \text{ is the frequence of the oscillation}$$

$$\mathbf{d}(x,y,z) = \begin{bmatrix} \begin{cases} 0.5(1 + \cos\left(2(x - 2.3)\pi + \pi\right)) * A * \sin\left(\omega t\right) + x\,, & \text{if } 2.29 < x < 2.8 \\ A * \sin(\omega t) + x\,, & \text{if } 2.8 < x < 3.8 \\ 0.5(1 + \cos(2(x - 2.3)\pi + \pi)) * A * \sin(\omega t) + x\,, & \text{if } 3.8 < x < 4.3 \end{cases} \\ y \\ z \end{bmatrix} . \tag{4.9}$$

The moved region is illustrated in Figure 4.6.

Figure 4.6: Bump extruded mesh with colored moved region.

Cost function

The output functional of interest is the quadratic deviation from ambient pressure on the surface S represented in Figure 4.7:

$$j(W) = \int_0^T \int_S \frac{1}{2}(p(t) - p_{air})^2 \mathrm{d}S\mathrm{d}t.$$

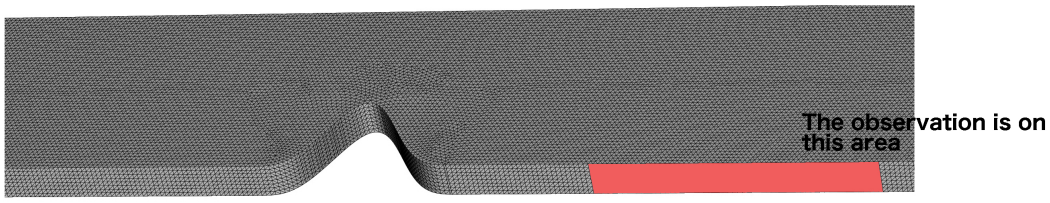


Figure 4.7: Surface area of interest.

The simulations were done in two dimensions and three dimensions(2D mesh extruded). As it is not possible to exactly compare the 2D solutions and 3D solutions because the physical phenomena are not the same in 2D and 3D (see Figure 4.8) the choice made is to compare the adjoint solution in the moving mesh case using the function $\mathbf{d}$ described in Equation (5.14) and the non moving mesh case. Results are briefly shown in Figure 4.9 and seem promising.

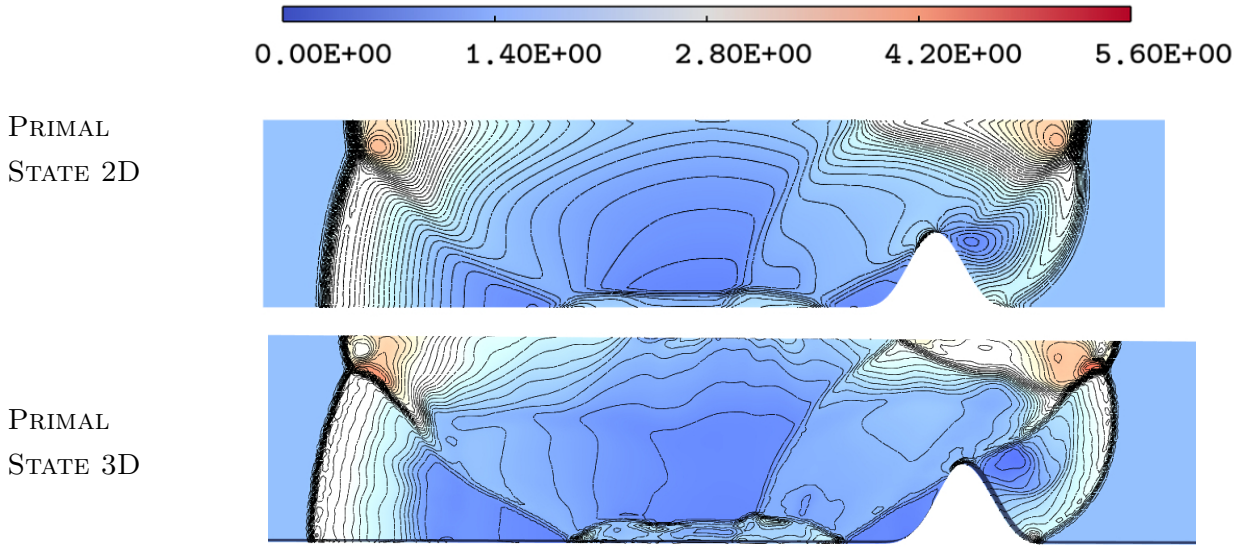


Figure 4.8: Blast with bump 2D/3D : Density at t=5.

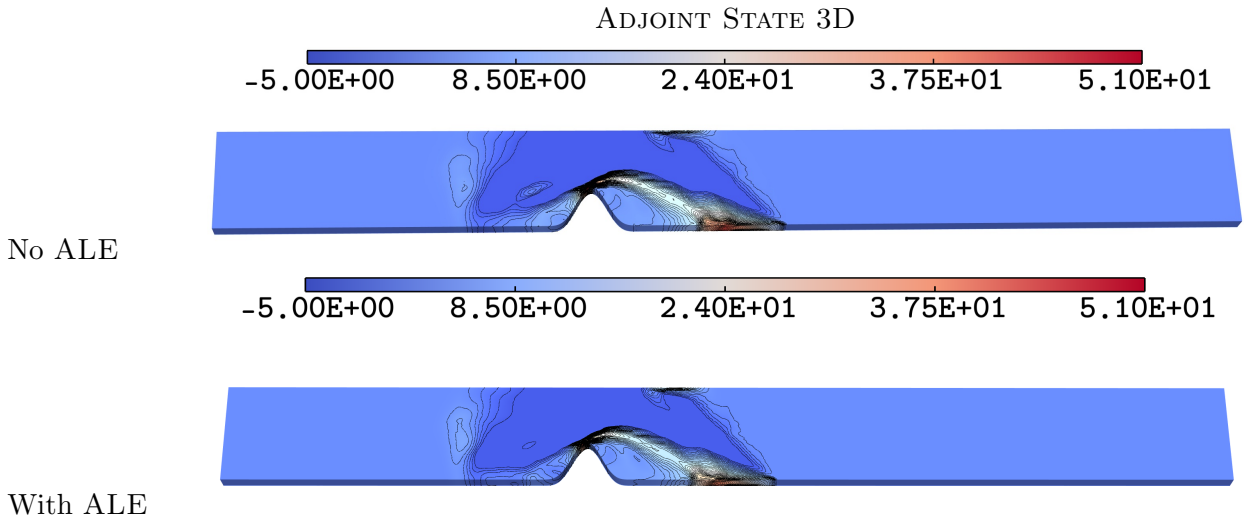


Figure 4.9: Blast with bump 3D : Adjoint of pressure at t=5.(Top) No ALE (Bottom) with ALE

4.9 Conclusion

The discrete adjoint system associated to unsteady inviscid Euler flows computed with an ALE formulation was presented for both explicit and implicit time integration schemes. Considering the adjoint state context rises new difficulties. Indeed, the unsteady adjoint state has to be

computed on a moving mesh and the mesh has to be moved backward in time. We need to introduce mesh velocities in the ALE adjoint state formulation. Therefore, time properties for backward moving mesh simulation were detailed. The opposite mesh velocity of forward moving mesh computation is considered to make backward meshes consistent with forward meshes. Thus, we have to address DGCL-like condition for the adjoint state introducing mesh velocities in the ALE adjoint state formulation. And numerical Jacobians for Riemann solvers have been presented.

In order to handle connectivity-changes a choice has been made to preserve memory. Each mesh optimization in the forward computation is tagged in order to make an optimization of the mesh at the same iteration in the backward moving-mesh computation. A validation of the adjoint has been performed to check the consistance of the jacobian fluxes.

Finally numerical cases were presented in 2D for an imposed displacement or for a fluid-structure interaction case. A 3D case of validation with comparison between moving mesh case and not moving mesh case have also been presented.

Chapter 5

Goal-oriented mesh adaptation for moving mesh simulations

When dealing with CFD problems, mesh adaptation is interesting for its ability to approach the asymptotic convergence and to obtain an accurate prediction for complex flows at a lower cost. Anisotropic mesh adaptation method reduces the number of degrees of freedom required to reach a given solution accuracy, thus impact favorably the CPU time. Moreover, it reduces the numerical scheme dissipation by automatically taking into account the anisotropy of the physical phenomena inside the mesh. Two main approaches of mesh adaptations exist in the literature. We have already seen the feature-based mesh adaptation in Chapter 2. Feature-based mesh adaptation is mainly deduced from an interpolation error estimate using the Hessian of the chosen sensor. It controls the interpolation error of the sensor over the whole computational domain. Such approach is easy to set-up and has a wide range of application, but it does not take into account the considered PDE used to solve the problem. If we want to focus on a particular output functional g, as the drag or the lift for example, we need to use a method of mesh optimisation which takes g into account. This is the role of the goal-oriented mesh adaptation. However, considering a simulation with moving geometries in a goal-oriented mesh adaptation context rises new difficulties. Indeed, the unsteady adjoint state has to be computed on a moving mesh and the mesh has to be moved backward in time when computing the unsteady adjoint state. For more explanations on these, see Chapter 4. Goal-oriented methods result from a series of papers dealing with *a posteriori* estimates as [Becker 1996, Giles 2002, Wintzer 2008]. But, extracting informations concerning *a posteriori* estimate is a difficult task. Starting from *a priori* estimates, Loseille *et al.* proposed a fully anisotropic goal-oriented mesh adaptation technique for steady problems [Loseille 2010b]. The authors combine goal-oriented rationale and application of Hessian-based analysis to truncation error. Then the study is extended to unsteady problems in [Belme 2012]. The error does not take into account the temporal error but concentrate only on spatial error for unsteady simulations.

In this present thesis, we combine fully anisotropic goal-oriented method of [Belme 2012] and

the fixed point advances of [N. Barral 2017] for time-accurate mesh adaptation involving moving geometries.

We start this chapter with a formal description of the error analysis in its most general expression, then the application to unsteady compressible Euler flows is presented. In Section 5.1, we introduce the optimal adjoint-based metric definition and all its relative issues, then Section 5.2 is dedicated to the implementation of the algorithm mesh optimization. This chapter ends with Section 5.3 and numerical experiments for blast wave problems.

5.1 Optimal adjoint-based ALE metric

We are interested in a scalar functional. The functions $W, W_h \in L^2(\Omega)$ are assumed to be the solution of the nonlinear continuous and discrete unsteady Euler PDE :

$$\Psi(\mathbf{W}) = 0 \quad ; \quad \Psi_h(\mathbf{W}_h) = 0$$

where :

$$\Psi(W) = W_t + \nabla \cdot \mathcal{F}_{ale}(W) = 0. \tag{5.1}$$

5.1.1 Functional output of interest

Let j be a smooth linear functional applied to W into the scalar number :

$$j(W) = (g, W)_{L^2(\Omega)} \quad ; \quad j(W_h) = (g, W_h)_{L^2(\Omega)}.$$

where (g, W) holds for the following rather most general functional output formulation :

$$(g, W) = \int_0^T \left[\int_\Omega (g_\Omega, W) \mathrm{d}\Omega + \int_\Gamma (g_\Gamma, W) \mathrm{d}\Gamma \right] \mathrm{d}t$$

The function g is a given $L^2(\Omega)$ function. And we are interested in minimizing the approximation error committed on the evaluation δj of j :

$$\delta j = j(W) - j(W_h)$$

5.1.2 Continuous adjoint state

Let us introduce the continuous and discrete systems in their variational formulation :

$$\text{Find } W \in \mathcal{V} \,|\, \forall \varphi \in \mathcal{V},\ (\Psi(W), \varphi) = 0,$$

$$\text{Find } W_h \in \mathcal{V}_h \,|\, \forall \varphi_h \in \mathcal{V}_h,\ (\Psi_h(W_h), \varphi_h) = 0.$$

The continuous and discrete adjoint systems in their variational formuation are:

$$\text{Find}\, W^* \in \mathcal{V}\,|\,\forall \psi \in \mathcal{V},\ \left(\frac{\partial \Psi}{\partial W}(W)\psi, W^*\right) = (g,\psi),$$

$$\text{Find}\, W_h^* \in \mathcal{V}_h\,|\,\forall \psi_h \in \mathcal{V}_h,\ \left(\frac{\partial \Psi_h}{\partial W}(W_h)\psi_h, W_h^*\right) = (g,\psi_h).$$

An integration by parts gives :

$$\begin{aligned}\left(\frac{\partial \Psi}{\partial W}(W)\psi, W^*\right) &= \int_{\Omega(t)} (\psi(0)W^*(0) - \psi(T)W^*(T))\ \mathrm{d}\Omega \\ &+ \int_0^T \int_{\Omega(t)} \psi \left(-W_t^* - \left(\frac{\partial \mathcal{F}_{ale}}{\partial W}\right)^* \nabla W^*\right) \mathrm{d}\Omega\, \mathrm{d}t \\ &+ \int_0^T \int_{\Gamma(t)} \psi \left[\left(\frac{\partial \mathcal{F}_{ale}}{\partial W}\right)^* W^*.\mathbf{n} - \left(\frac{\partial \hat{\mathcal{F}}_{ale}}{\partial W}\right)^* W^*.\mathbf{n}\right] \mathrm{d}\Gamma\, \mathrm{d}t\ .\end{aligned}$$

Consequently, the continuous adjoint state W^* must be such that:

$$-W_t^* - \left(\frac{\partial \mathcal{F}_{ale}}{\partial W}\right)^* \nabla W^* = g_{\Omega(t)} \text{ in } \Omega(t), \tag{5.2}$$

with the associated adjoint boundary conditions:

$$\left(\frac{\partial \mathcal{F}_{ale}}{\partial W}\right)^* W^*.\mathbf{n} - \left(\frac{\partial \hat{\mathcal{F}}_{ale}}{\partial W}\right)^* W^*.\mathbf{n} = g_{\Gamma(t)} \text{ on } \Gamma(t), \tag{5.3}$$

and the final adjoint state condition:

$$W^*(T) = g_T.$$

The adjoint Euler equations is also a system of advection equations, where the temporal integration goes backward.

5.1.3 Formal error analysis

The idea is now to compute the difference of variational residual for a discrete test function:

$$\delta j = j(W) - j(W_h) = (g, W) - (g, W_h),$$

$$(\Psi_h(W), \varphi_h) - (\Psi_h(W_h), \varphi_h) = (\Psi_h(W) - \Psi(W), \varphi_h)\,.$$

Then assuming that W^*, $\Pi_h W^*$ and W_h^* and their gradients are close to each other:

$$\delta j \approx (\Psi_h(W) - \Psi(W), W^*) \approx (\Psi_h(W) - \Psi(W), \Pi_h W^*)\,. \tag{5.4}$$

The term $\Psi_h(W) - \Psi(W)$ is an *a posteriori* local error which we now evaluate.

5.1.4 Local error analysis

We replace in Estimation (5.4) operators Ψ and Ψ_h by their expressions given by Relation (5.1). We also discard the error committed when imposing the initial condition. We finally get the following simplified error model:

$$\begin{aligned}
\delta j \;\approx\; & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^{n+1}}} \Pi_h W^{*,n+1}(\Pi_h W^{n+1} - W^{n+1})\,\mathrm{d}\Omega \\
- & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^n}} \Pi_h W^{*,n}(\Pi_h W^{n} - W^{n})\,\mathrm{d}\Omega \\
+ & \sum_{n=1}^{n=nmax} \int_{\Omega_{t^n}} \Pi_h W^{*,n}\,\nabla\cdot(\Pi_h \mathcal{F}_{ale}(W^n) - \mathcal{F}_{ale}(W^n))\,\mathrm{d}\Omega \\
- & \sum_{n=1}^{n=nmax} \int_{\partial\Omega_{t^n}} \Pi_h W^{*,n}\,(\Pi_h \hat{\mathcal{F}}_{ale}(W^n) - \hat{\mathcal{F}}_{ale}(W^n)).\mathbf{n}\,\mathrm{d}\Gamma.
\end{aligned} \tag{5.5}$$

Integrating by parts leads to:

$$\begin{aligned}
\delta j \;\approx\; & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^{n+1}}} \Pi_h W^{*,n+1}(\Pi_h W^{n+1} - W^{n+1})\,\mathrm{d}\Omega \\
- & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^n}} \Pi_h W^{*,n}(\Pi_h W^{n} - W^{n})\,\mathrm{d}\Omega \\
- & \sum_{n=1}^{n=nmax} \int_{\Omega_{t^n}} \nabla\Pi_h W^{*,n}\,\cdot(\Pi_h \mathcal{F}_{ale}(W^n) - \mathcal{F}_{ale}(W^n))\,\mathrm{d}\Omega \\
- & \sum_{n=1}^{n=nmax} \int_{\partial\Omega_{t^n}} \Pi_h W^{*,n}\,(\Pi_h \bar{\mathcal{F}}_{ale}(W^n) - \bar{\mathcal{F}}_{ale}(W^n)).\mathbf{n}\,\mathrm{d}\Gamma.
\end{aligned} \tag{5.6}$$

with $\bar{\mathcal{F}}_{ale} = \hat{\mathcal{F}}_{ale} - \mathcal{F}_{ale}$. We observe that this estimate of δj is expressed in terms of interpolation errors of the Euler fluxes and of the time derivative weighted by continuous functions $\Pi_h W^* \approx W^*$ and $\nabla\Pi_h W^* \approx \nabla W^*$. The integrands in Error Estimation (5.6) contain positive and negative parts which can compensate for some particular meshes. In our strategy, we prefer not to rely on these parasitic effects and to slightly over-estimate the error. To this end, all

integrands are bounded by their absolute values:

$$\begin{aligned}
\delta j \quad \leq \quad & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^{n+1}}} |\Pi_h W^{*,n+1}||\Pi_h W^{n+1} - W^{n+1}| \, \mathrm{d}\Omega \\
+ \quad & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^n}} |\Pi_h W^{*,n}||\Pi_h W^{n} - W^{n}| \, \mathrm{d}\Omega \\
+ \quad & \sum_{n=1}^{n=nmax} \int_{\Omega_{t^n}} |\nabla \Pi_h W^{*,n}| \; |\Pi_h \mathcal{F}_{ale}(W^n) - \mathcal{F}_{ale}(W^n)| \, \mathrm{d}\Omega \\
+ \quad & \sum_{n=1}^{n=nmax} \int_{\partial\Omega_{t^n}} |\Pi_h W^{*,n}| \, |(\Pi_h \bar{\mathcal{F}}_{ale}(W^n) - \bar{\mathcal{F}}_{ale}(W^n)).\mathbf{n}| \, \mathrm{d}\Gamma. \qquad (5.7)
\end{aligned}$$

5.1.5 Continuous error model

Working in the continuous framework enables to write Estimate (5.7) in a *spatially*-continuous form, in which the Π_h are discarded, and in which the interpolation error $Id - \Pi_h$ is replaced by its continuous $Id - \pi_{\mathcal{M}}$. Then, we are interested in minimizing the following error functional:

$$\begin{aligned}
\mathbf{E}(\mathcal{M}) \quad = \quad & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^{n+1}}} |W^{*,n+1}||\pi_{\mathcal{M}} W^{n+1} - W^{n+1}| \, \mathrm{d}\Omega \\
+ \quad & \sum_{n=1}^{n=nmax} \frac{1}{t^{n+1}-t^n} \int_{\Omega_{t^n}} |W^{*,n}||\pi_{\mathcal{M}} W^{n} - W^{n}| \, \mathrm{d}\Omega \\
+ \quad & \sum_{n=1}^{n=nmax} \int_{\Omega_{t^n}} |\nabla W^{*,n}| \; |\pi_{\mathcal{M}} \mathcal{F}_{ale}(W^n) - \mathcal{F}_{ale}(W^n)| \, \mathrm{d}\Omega \\
+ \quad & \sum_{n=1}^{n=nmax} \int_{\partial\Omega_{t^n}} |W^{*,n}| \, |(\pi_{\mathcal{M}} \bar{\mathcal{F}}_{ale}(W^n) - \bar{\mathcal{F}}_{ale}(W^n)).\mathbf{n}| \, \mathrm{d}\Gamma. \qquad (5.8)
\end{aligned}$$

We observe that the fourth term introduces a dependency of the error with respect to the boundary surface mesh. In the present paper, we discard this term and refer to [Loseille 2010a] for a discussion on its influence. The first term can be transformed as follows without introducing a large error:

$$\int_{\Omega_{t^{n+1}}} |W^{*,n+1}||\pi_{\mathcal{M}} W^{n+1} - W^{n+1}| \, \mathrm{d}\Omega = \int_{\Omega_{t^n}} |J_n^{n+1}|^{-1} |\hat{W}^{*,n+1}||\pi_{\mathcal{M}} \hat{W}^{n+1} - \hat{W}^{n+1}| \, \mathrm{d}\Omega$$

where $|J_n^{n+1}|$ is the determinant of the transformation from Ω_n to Ω_{n+1},

$$J_n^{n+1} = det \frac{\partial \phi(., t^{n+1}) \circ \phi^{-1}(., t^n)}{\partial \mathbf{x}} \qquad (5.9)$$

and $\hat{W}^{*,n+1}$ resp. $\hat{W}^{n+1}$ the functions of Ω_n obtained by reverse transportation from Ω_{n+1}:

$$\begin{aligned}
\forall \mathbf{x} \in \Omega_{t^n} \quad & \hat{W}^{n+1}(\mathbf{x}) = W^{n+1}\left(\phi(\phi^{-1}(\mathbf{x}, t^n), t^{n+1}\right), \\
\forall \mathbf{x} \in \Omega_{t^n} \quad & \hat{W}^{*,n+1}(\mathbf{x}) = W^{*,n+1}\left(\phi(\phi^{-1}(\mathbf{x}, t^n), t^{n+1}\right). \qquad (5.10)
\end{aligned}$$

Then, introducing the continuous interpolation error, we can write the simplified error model as follows:

$$\mathbf{E}(\mathbf{M}) \;=\; \sum_{n=1}^{n=nmax} \int_{\Omega} \mathrm{trace}\left(\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t^n)\,\mathbf{H}_{go}^{ALE}(\mathbf{x},t^n)\,\mathcal{M}^{-\frac{1}{2}}(\mathbf{x},t^n)\right)\,\mathrm{d}\Omega\,\mathrm{d}t$$

with $\mathbf{H}_{go}^{ALE}(\mathbf{x},t^n) = \sum_{j=1}^{5} \mathbf{H}_{ale,j}^{GO}(\mathbf{x},t^n)$, in which

$$\begin{aligned}
\mathbf{H}_{j,go}^{ALE}(\mathbf{x},t^n) \;&=\; \frac{1}{t^{n+1}-t^n}\left||J_n^{n+1}|^{-1}\hat{W}_j^{*,n+1}(\mathbf{x},t)\right|\cdot\left|H(\hat{W}_j^{n+1})(\mathbf{x})\right| \\
&+\; \frac{1}{t^{n+1}-t^n}\left|W_j^{*,n}(\mathbf{x})\right|\cdot\left|H(W_j^n)(\mathbf{x},t)\right| \\
&+\; \left|\frac{\partial W_j^{*,n}}{\partial x}(\mathbf{x})\right|\cdot\left|H(\mathcal{F}_{ale,1}(W_j^n))(\mathbf{x})\right| \;+\; \left|\frac{\partial W_j^{*,n}}{\partial y}(\mathbf{x})\right|\cdot\left|H(\mathcal{F}_{ale,2}(W_j^n))(\mathbf{x})\right| \\
&+\; \left|\frac{\partial W_j^{*,n}}{\partial z}(\mathbf{x})\right|\cdot\left|H(\mathcal{F}_{ale,3}(W_j^n))(\mathbf{x})\right|
\end{aligned} \tag{5.11}$$

is defined on Ω_{t^n}. Here, W_j^* denotes the j^{th} component of the adjoint vector W^*, $H(\mathcal{F}_{ale,i}(W_j))$ the Hessian of the j^{th} component of the vector $\mathcal{F}_{ale,i}(W)$, and $H(W_{j,t})$ the Hessian of the j^{th} component of the time derivative of W.
We observe that the first line of RHS is a time derivative, since :

$$\begin{aligned}
&\frac{1}{t^{n+1}-t^n}\left||J_n^{n+1}|^{-1}\hat{W}_j^{*,n+1}(\mathbf{x},t)\right|\cdot\left|H(\hat{W}_j^{n+1})(\mathbf{x})\right| - \left|W_j^{*,n}(\mathbf{x})\right|\cdot\left|H(W_j^n)(\mathbf{x},t)\right| \\
=\; &\frac{1}{t^{n+1}-t^n}\left||J_n^{n+1}|^{-1}\hat{W}_j^{*,n+1}(\mathbf{x},t)\right|\cdot\left|H(\hat{W}_j^{n+1})(\mathbf{x})\right| - \left||J_n^{n}|^{-1}\hat{W}_j^{*,n}(\mathbf{x})\right|\cdot\left|H(\hat{W}_j^{n})(\mathbf{x},t)\right|
\end{aligned} \tag{5.12}$$

where:

$$\begin{aligned}
\forall \mathbf{x}\in\Omega_{t^n}\quad \hat{W}^n(\mathbf{x}) &= W^n\left(\phi(\phi^{-1}(\mathbf{x},t^n),t^n\right), \\
\forall \mathbf{x}\in\Omega_{t^n}\quad \hat{W}^{*,n}(\mathbf{x}) &= W^{*,n}\left(\phi(\phi^{-1}(\mathbf{x},t^n),t^n\right).
\end{aligned} \tag{5.13}$$

5.2 Implementation of goal-oriented mesh adaptation for unsteady problems involving moving geometries

5.2.1 Choice of the goal-oriented metric

The optimal adapted meshes for each sub-interval are generated according to the previous analysis. In this work, the following particular choices have been made :

- the Hessian metric for sub-interval i is based on a control of the temporal error in $\mathbf{L}^1$ norm
- function $\tau : t \to \tau(t)$ is constant and equal to 1

- all sub-intervals have the same time length Δt.

The optimal goal-oriented metric simplifies to :

$$\mathcal{M}_{go}^{i,ALE}(\mathbf{x})$$
$$= \mathcal{N}_{st}^{\frac{2}{n}} \left(\sum_{j=1}^{n_{adap}} \mathcal{K}^{j,ALE} \left(\int_{t_{j-1}}^{t_j} \tau(t)^{-1} \mathrm{d}t \right)^{\frac{2p}{2p+n}} \right)^{-\frac{2}{n}} \left(\int_{t_{i-1}}^{t_i} \tau(t)^{-1} \mathrm{d}t \right)^{-\frac{2}{2p+n}} (\det \mathbf{H}_{go}^{i,ALE}(\mathbf{x}))^{-\frac{1}{2p+n}} \mathbf{H}_{go}^{i,ALE}(\mathbf{x})$$

$$\text{where } \mathcal{K}^{j,ALE} = \left(\int_\Omega \left(\det \mathbf{H}_{go}^{j,ALE}(\mathbf{x}) \right)^{\frac{p}{2p+n}} \mathrm{d}\mathbf{x} \right)$$

Remark 5. *We notice that we obtain a similar expression of the optimal metric to those presented in Chapter 1.*

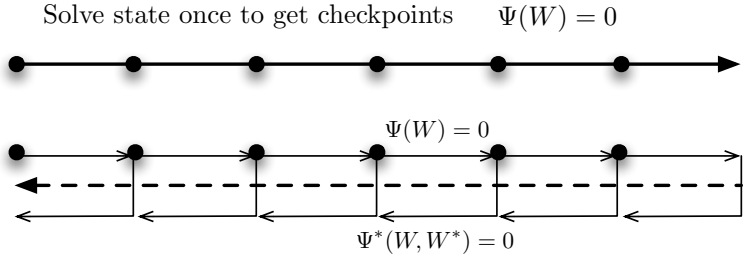
5.2.2 Memory management

The main difference with the feature-based fixed-point mesh adaptation lies in the backward computation of the unsteady adjoint state which requires to know the solution on the whole simulation interval [0,T].

As seen in Chapter 4, the memory effort can be reduced by out-of-core storage of checkpoints as shown in the picture below. First the state-simulation is performed to store checkpoints. Second, when computing backward the adjoint, we first recompute all states from the checkpoint and store them in memory and then we compute the unsteady adjoint until the checkpoint physical time. This method implies a recomputing effort of the state W.

As the fixed-point mesh adaptation algorithm split the time frame into n_{adap} sub-intervals for mesh adaptation, this sub-intervals are used as checkpoints for the adjoint computation.

If the computation of the adjoint states need to much memory storage, the other strategy consists in storing solution states in memory only each m solver iterations. When the unsteady adjoint is solved, solution states between two savings are linearly interpolated. So the process is the following : at first, no interpolation is done, then if the system complains about a lack of memory, one solution out of two is deleted and interpolations are done and so on : if the memory capacity is still insufficient, only one solution out of four is kept and interpolation is made etc. This method leads inevitably to a loss of accuracy for the unsteady adjoint computation.



Solve state and backward adjoint state from checkpoints

5.2.3 Algorithm

The Algorithm 11 is complicated by the fact that the unsteady adjoint solver must be computed backward in time (eg. step 2) after computing the solution over the simulation time frame.

Algorithm 11 Goal-Oriented Mesh Adaptation for Unsteady Flows

---

Input : Initial mesh and solution $(\mathcal{H}_0, \mathcal{S}_0^0)$ and set targeted space-time complexity $\mathcal{N}_{st}$

 # Fixed-point loop to converge the global space-time goal-oriented mesh adaptation problem

For $j = 1, n_{ptfx}$

 # Adaptive loop to compute forward the solution state in time on time frame $[0, T]$

1. **For** $i = 1, n_{adap}$

   (a) $\mathcal{S}_{0,i}^j$ = Interpolate conservatively next sub-interval initial solution from $(\mathcal{H}_{i-1}^j, \mathcal{S}_{i-1}^j, \mathcal{H}_i^j)$;

   (b) $\mathcal{S}_i^j$ = Compute solution on sub-interval from pair $(\mathcal{H}_i^j, \mathcal{S}_{0,i}^j)$;

   EndFor

 # Adaptive loop to compute backward the adjoint state in time on time frame $[T, 0]$

2. **For** $i = n_{adap}, 1$

   (a) $(\mathcal{S}^*)_i^j$ = Interpolate previous sub-interval final adjoint state from $(\mathcal{H}_i^j, \mathcal{H}_{i+1}^j, (\mathcal{S}_0^*)_{i+1}^j)$;

   (b) $(\mathcal{S}_0^*)_i^j$ = Compute backward adjoint state on sub-interval from $(\mathcal{H}_i^j, \mathcal{S}_i^j, (\mathcal{S}^*)_i^j)$;

   (c) $|\mathbf{H}_{go,L^1}|_i^j$ = Compute sub-interval goal-oriented Hessian-metric from sample $(\mathcal{H}_i^j, \{\mathcal{S}_i^j(k), (\mathcal{S}^*)_i^j(k)\}_{k=1}^{nk})$;

   EndFor

3. $\mathcal{C}^j$ = Compute space-time complexity from all goal-oriented Hessian-metrics $\{|\mathbf{H}_{go,L^1}|_i^j\}_{i=1}^{n_{adap}}$;

4. $\{\mathcal{M}_i^j\}_{i=1}^{n_{adap}}$ = Compute all sub-interval unsteady metrics $(\mathcal{C}^j, \{|\mathbf{H}_{go,L^1}|_i^j\}_{i=1}^{n_{adap}})$;

5. $\{\mathcal{H}_i^{j+1}\}_{i=1}^{n_{adap}}$ = Generate all sub-interval adapted meshes $(\{\mathcal{H}_i^j,\ \mathcal{M}_i^j\}_{i=1}^{n_{adap}})$;

EndFor

5.3 Numerical test cases

We present in this section some preliminary results of our goal-oriented mesh adaptation algorithm for an unsteady simulation with moving domain.

5.3.1 Numerical test case : a Lohner blast-like on a 2D Bump

The test case presented is a Löhner semi-circular blast-like simulation in a domain $[0,5]\times[0,0.5]$ with a bump centered in $(1.875,0)$ of diameter 0.4. The circular blast is located on $(1.2,0)$. At initial time, the state is $(\rho_{blast},\mathbf{u}_{blast},p_{blast})=(10,\mathbf{0},25)$ whereas for the rest of the domain the state is $(\rho_{domain},\mathbf{u}_{domain},p_{domain})=(1,\mathbf{0},2.5)$.

Moving mesh

The moving mesh function is :

$$A=0.15 \text{ is the amplitude of the oscillation}$$
$$\omega=\pi/0.7 \text{ is the frequence of the oscillation}$$

$$\mathbf{d}(x,y) = \begin{bmatrix} \begin{cases} 0.5(1+\cos(2(x-2.3)\pi+\pi))*A*\sin(\omega t)+x\,, & \text{if } 2.29<x<2.8 \\ A*\sin(\omega t)+x\,, & \text{if } 2.8<x<3.8 \\ 0.5(1+\cos(2(x-2.3)\pi+\pi))*A*\sin(\omega t)+x\,, & \text{if } 3.8<x<4.3 \end{cases} \\ \\ y \end{bmatrix}. \tag{5.14}$$

The moving region is illustrated in Figure 5.1 (Top). It is represented by the red triangles. The blue ones are fixed.

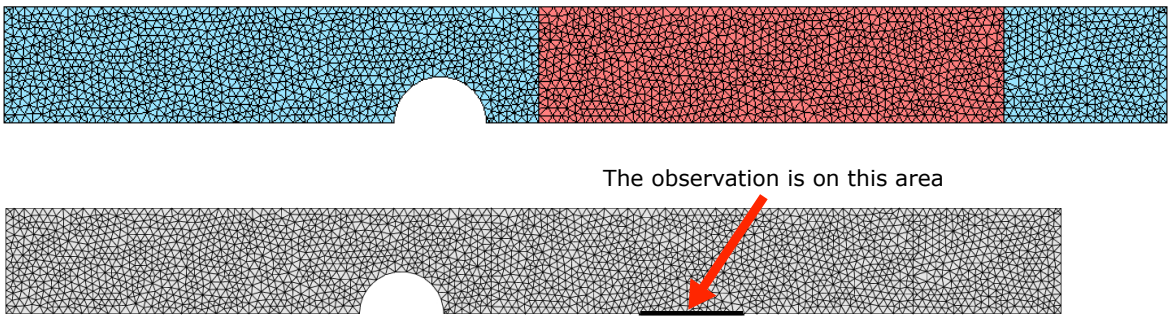


Figure 5.1: (Top)The region in blue is fixed while the red region moves. (Bottom) Illustration of the output functional observation area

Cost function

The output functional is the quadratic deviation from ambient pressure on the surface S represented in Figure 5.1 (bottom):

$$j(W) = \int_0^T \int_S \frac{1}{2}(p(t) - p_{air})^2 \mathrm{d}S\mathrm{d}t.$$

5.3.2 Numerical results

Two simulations are performed. The first one considers the Hessian-based mesh adaptation coupled with moving geometries presented in Chapter 3 and the second one uses the Goal-oriented mesh adaptation coupled with moving geometries presented in this chapter. The considered displacement is smooth as no shearing occurs which might require connectivy-changes. Thus no connectivity-changes are done during the simulation.

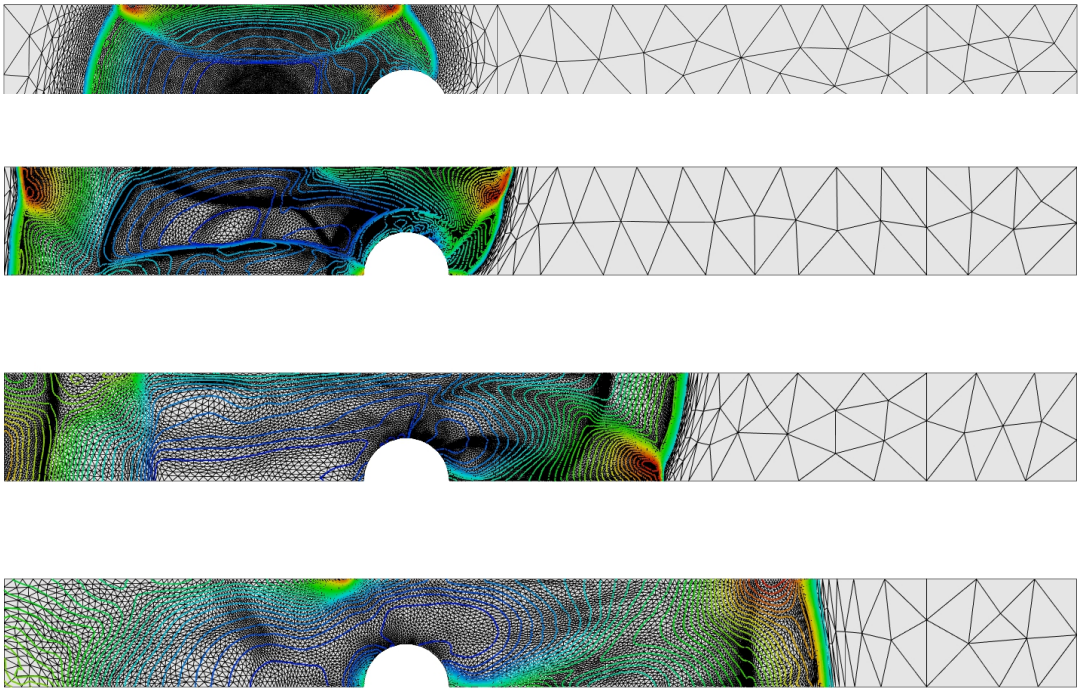


Figure 5.2: Hessian-based mesh adaptation

In goal-oriented mesh adaptation the mesh is adapted only on the flow interesting for the functional output while in hess adaptation, the remeshing is not quite dense.

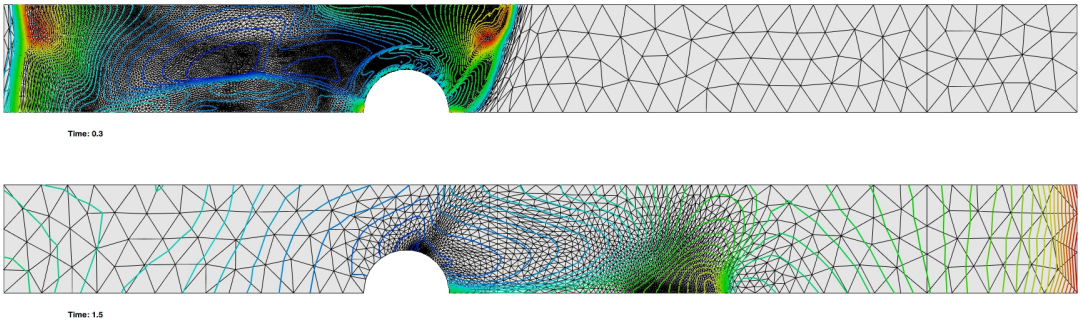


Figure 5.3: Goal-oriented mesh adaptation

5.4 Conclusion

This chapter addresses goal-oriented mesh adaptation for time dependent and dynamic meshes simulations. In the first section, the error analysis is widely detailed. In the second section, the choice of metric optimization is presented. This concept is used to extend the previous unsteady goal-oriented mesh adaptation of [Belme 2011] to moving geometries.

The algorithm follows the work of [Olivier 2011a] and [Barral 2015] to generate meshes that are adapted once moved with a certain displacement. Analytic examples validated this metric, and it was used within a new error analysis, to derive an ALE metric for adaptive sub-intervals. The regular time-dependent algorithm was modified to use this metric, and coupled with the moving mesh algorithm and example is done on a 2D analytic example.

The main contributions of this thesis with respect to pre-existing work in the GAMMA group is:

- An update strategy was proposed for the metric at mesh optimization times, to avoid spoiling the adaptation with optimizations.
- A study of the error on analytic case for the ALE goal-oriented metric that confirmed its adapted characteristic.
- A new version of the global fixed-point unsteady mesh adaptation algorithm was used.

Several perspectives arise from this work, notably:

- To improve the adaptation algorithm, the next step is to control the temporal error. This would in particular lead to automatic adaptation of the size of the sub-intervals. This appears to be all the more important with moving meshes, where the size of the mesh is not even constant throughout a sub-interval and time step. This improvement is necessary to deal with the ALE implicit scheme presented in Chapter 2.

- The convergence of this unsteady adaptive method needs to be studied, depending on the number of vertices and/or the size of the sub-intervals, as well as the influence of the initial solution on the final solution, and the ability of the algorithm to recover features of the solution lost due to coarse initial meshes.

Conclusions and Perspectives

Conclusions

This thesis was carried out at Inria within the teams GAMMA3 (Inria Saclay) and ECUADOR (Inria Sophia-Antipolis). It has been part of the continuous works on mesh adaptation of these two teams and contributes to the developpement of the numerical in-house fluid solver `Wolf`. This thesis has presented several novelties regarding the extension of both hessian-based and goal-oriented metric-based anisoltropic mesh adaptation to unsteady simulations for moving computational domains.
The problematics raised by mesh adaptation, mesh movement and adjoint method have been studied and analyzed at all the levels of the numerical resolution : the primal and dual solver phases, the forward and backward moving mesh algorithms and the mesh adaptation.

The work presented in this thesis follows from [Olivier 2011a, Belme 2011, Barral 2015] and presented updates for most stages of the resolution loop. Aside from the important theorical and technical prerequisites required for this work, the challenge was triple. To compute mesh adaptation for moving computational domains, the moving mesh must be performed with precision, then intelligent solutions must be performed to compute the adjoint solver backward in ALE formulation on a moving mesh and make it consistent with the flowfield state. And the last difficulty is to preserve the mesh quality after remeshing. For that reason, the mesh adaptation algorithm is divided into sub-intervals and coupled with an optimization algortihm of connectivity changes.

The main contribution on this thesis concerns the adjoint solver and the goal-oriented mesh adaptation. A new space-time goal-oriented error estimator has been designed to handle moving mesh simulations. It has been derived from the unsteady Euler equation in the ALE formulation. This ALE goal-oriented metric takes the movement of the mesh into account in the error estimate. Moreover the consideration of the mesh optimization made after the remeshing and during the connectivity change loop ensures the accuracy and the mesh quality of the ALE-mesh adaptation both for hessian-based mesh adaptation presented in Chapter 3 and for goal-oriented mesh adaptation in Chapter 5. The ALE solver has been adapted to handle this new backward moving mesh framework. In fact, this work also presents an implementation of adjoint solver for time dependent simulations with moving meshes in

Chapter 4. This solver allows in particular mesh connectivity changes. Therefore, the backward changing-connectivity ALE scheme has been developed and validated in two dimensions and introduced in three dimensions. Finally and in the perspective of a space-time mesh adaptation, implicit solvers in ALE formulation for moving geometries problems have been developed and implemented in three-dimensional moving mesh simulations as illustrated in Chapter 2. It is also already effective in terms of CPU time as compared to explicit time integration solvers.

Perspectives

A wide variety of perspectives emerge from this work. We list there below :

- The first natural one is the validation of 3D adjoint solver in ALE formulation and computation for more complex simulations. Then extend the goal-oriented mesh adaptation for 3D simulations.
- Since only rigid bodies are considered in this thesis, one perspective could be to consider deformable bodies.
- In the field of mesh deformation, other kinds of moving mesh equations can be investigated. A premice of this investigation can be find in [Barral 2015].
- Future work will certainly focus on improving the space-time error analysis with moving meshes, notably by introducing a better control of the temporal error. This would notably result in an automatic adaptation of the sub-intervals sizes of the fixed-point algorithm. Combined with the implicit time-integration solver in ALE formulation presented in this thesis could improve the simulation time significantly.
- In the future, extending this work to goal-oriented mesh adaptation for Navier-Stokes equations will lead to a greatest interest for the method to industrials. Already, a lot of ALE schemes are available for viscous simulations [Hay 2014], which facilitate the task, together with many works on adaptation for viscous flows [Menier 2015, Frazza 2018].
- Time consumption is a huge limitation in the size and the complexity of the problems considered. Mesh adaptation appears as an answer to this problem, by helping to reduce greatly the size of the meshes while preserving the solution accuracy. However, it is only really efficient if it is coupled to an efficient parallelization of the codes. In this thesis, only shared memory parallelization has been considered. In the medium term, it seems that considering other kinds of parallelization will be unavoidable: reflections on parallelization of meshing software on distributed memory were started in [Loseille 2013], and the parallelization of the

fluid solver on GPU was started in [Barral 2015]. Both approaches may have to be combined in order to take advantage of recent hybrid computation clusters.

- The FSI aspect can also be improved, by replacing the 6-DOF model by continuous elastic materials. The loose coupling could be replaced by more sophisticated strong coupling schemes. In this way, complex problems from aeroelasticity to biology could be run, in the interest of both industries and researchers.

- Another promising research perspective for the future is the extension of metric-based mesh adaptation to curved meshes. In fact, high-order elements have been used for a long while, mostly in Finite Element Methods, but only by increasing the order of the approximation polynomials. It has not been followed by meshing software, who have been generating flat pseudo high-order elements until very recently, *i.e.* standard $\mathbb{P}^1$ elements with more nodes on the edges. The generation of really high-order meshes is only in its earliest infancy. Lately, progress has been made on the generation of $\mathbb{P}^2$ curved meshes [Johnen 2013, Abgrall 2014]. However, a smart use of curved meshes seems to be promising, notably by enabling a much more accurate representation of curved boundaries.

Bibliography

[A. Bueno-Orovio 2007] F. Palacios A. Bueno-Orovio C. Castro and E. Zuazua. *Continuous adjoint approach for the Spalart-Allmaras model in aerodynamic optimization.* AIAA Journal, vol. 45, no. 9, pages 2125–2139, 2007.

[A. Sei 1995] W. Symes A. Sei. *A note on consistency and adjointness for numerical schemes.* Technical Report TR95-04, Department of Computational and Applied Mathematics, Rice University, 1995.

[A. Tikhonov 1977] V. Arsenin A. Tikhonov. Solutions of ill-posed problems. V.H Winston & Sons, Washington, D.C., 1977.

[Abgrall 2014] R. Abgrall, C. Dobrzynski and A. Froehly. *A method for computing curved meshes via the linear elasticity analogy, application to fluid dynamics problems.* International Journal for Numerical Methods in Fluids, vol. 76, no. 4, 2014.

[Alauzet 2007] F. Alauzet, P.J. Frey, P.L. George and B. Mohammadi. *3D transient fixed point mesh adaptation for time-dependent problems: Application to CFD simulations.* J. Comp. Phys., vol. 222, pages 592–623, 2007.

[Alauzet 2009] F. Alauzet and A. Loseille. *On the use of space filling curves for parallel anisotropic mesh adaptation.* In Proceedings of the 18th International Meshing Roundtable, pages 337–357. Springer, 2009.

[Alauzet 2010a] F. Alauzet. *Size gradation control of anisotropic meshes.* Finite Elem. Anal. Des., vol. 46, pages 181–202, 2010.

[Alauzet 2010b] F. Alauzet and A. Loseille. *High Order Sonic Boom Modeling by Adaptive Methods.* J. Comp. Phys., vol. 229, pages 561–593, 2010.

[Alauzet 2010c] F. Alauzet and M. Mehrenberger. *P1-conservative solution interpolation on unstructured triangular meshes.* Int. J. Numer. Meth. Engng, vol. 84, no. 13, pages 1552–1588, 2010.

[Alauzet 2011a] F. Alauzet and G. Olivier. *Extension of Metric-Based Anisotropic Mesh Adaptation to Time-Dependent Problems Involving Moving Geometries.* In 49th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2011-0896, Orlando, FL, USA, Jan 2011.

[Alauzet 2011b] F. Alauzet and G. Olivier. *Extension of Metric-Based Anisotropic Mesh Adaptation to Time-Dependent Problems Involving Moving Geometries.* In 49th AIAA Aerospace Sciences Meeting, AIAA Paper 2011-0896, Orlando, FL, USA, Jan 2011.

[Alauzet 2012] F. Alauzet. Contributions aux méthodes numériques pour l'adaptation de maillage et le maillage mobile. Habilitation à Diriger des Recherches, Université Pierre et Marie Curie, Paris VI, Paris, France, 2012.

[Alauzet 2014] F. Alauzet. *A changing-topology moving mesh technique for large displacements.* Engineering with Computers, vol. 30, no. 2, pages 175–200, 2014.

[Alauzet 2016] F. Alauzet and A. Loseille. *A decade of progress on anisotropic mesh adaptation for Computational Fluid Dynamics.* Comput. Aided Des., vol. 72, pages 13–39, 2016. Accepted.

[Allaire 2007] G. Allaire. Conception optimale des structures. Springer, Berlin, 58 édition, 2007.

[Allaire 2015] G. Allaire. *A review of adjoint methods for sensitivity analysis, uncertainty quantification and optimization in numerical codes.* Ingénieurs de l'automobile, vol. 836, pages 33–36, 2015.

[Anderson 1997] K. Anderson and V. Venkatakrishnan. *Aerodynamic Design Optimization on Unstructured Grids with a Continuous Adjoint Formulation.* AIAA 35'th Aerospace Sciences Meeting and Exhibit, 1997.

[Arian 1997] E. Arian and D. Salas. *Admitting the Inadmissible: Adjoint Formulation for Incomplete Cost Functionals in Aerodynamic Optimization.* Research Report 97-69, ICASE, december 1997.

[Astorino 2009] M. Astorino, F. Chouly and M.A. Fernández. *Robin-based Semi-Implicit Coupling in Fluid-Structure Interaction: Stability Analysis and Numerics.* SIAM J. Sci. Comput., vol. 31, no. 6, pages 4041–4065, 2009.

[A.Stück 2009] A.Stück, F. Camelli and R. Löhner. *Adjoint-based design of shock mitigation devices.* International journal for numerical methods in fluids, 2009.

[B. Seny 2014] T. Toulorge V. Legat J.-F. Remacle B. Seny J. Lambrechts. *An efficient parallel implementation of explicit multirate Runge-Kutta schemes for discontinuous Galerkin computations.* Journal of Computational Physics, vol. 256, pages 135–160, 2014.

[Baines 1994] M.J. Baines. Moving finite elements. Oxford University Press, Inc., New York, NY, 1994.

[Baker 1999] T.J. Baker and P. Cavallo. *Dynamic adaptation for deforming tetrahedral meshes.* AIAA Journal, vol. 19, pages 2699–3253, 1999.

[Bank 1993] R.E. Bank and R.K. Smith. *A posteriori error estimate based on hierarchical bases.* SIAM J. Numer. Anal., vol. 30, pages 921–935, 1993.

[Barral 2015] N. Barral. *Time-accurate anisotropic mesh adaptation for three-dimensional moving mesh problems.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2015.

[Batina 1990] J. Batina. *Unsteady Euler Airfoil Solutions Using Unstructured Dynamic Meshes.* AIAA Journal, vol. 28, no. 8, pages 1381–1388, 1990.

[Batten 1997] P. Batten, N. Clarke, C. Lambert and D.M. Causon. *On the choice of wavespeeds for the HLLC Riemann solver.* SIAM J. Sci. Comput., vol. 18, no. 6, pages 1553–1570, 1997.

[Baum 1994] J.D. Baum, H. Luo and R. Löhner. *A New ALE Adaptive Unstructured Methodology for the Simulation of Moving Bodies.* In 32th AIAA Aerospace Sciences Meeting, AIAA Paper 1994-0414, Reno, NV, USA, Jan 1994.

[Baum 1996] J.D. Baum, H. Luo, R. Löhner, C. Yang, D. Pelessone and C. Charman. *A Coupled Fluid/Structure Modeling of Shock Interaction with a Truck.* In AIAA-96-0795, 1996.

[Bazilevs 2011] 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.* International Journal for Numerical Methods in Fluids, vol. 65, no. 1-3, pages 236–253, 2011.

[Becker 1996] R. Becker and R. Rannacher. *A feed-back approach to error control in finite element methods: basic analysis and examples.* East-West J. Numer. Math., vol. 4, pages 237–264, 1996.

[Belme 2011] A. Belme. *Unsteady Aerodynamics and Adjoint Method.* PhD thesis, Université de Nice Sophia-Antipolis, Nice, France, 2011.

[Belme 2012] A. Belme, A. Dervieux and F. Alauzet. *Time Accurate Anisotropic Goal-Oriented Mesh Adaptation for Unsteady Flows.* J. Comp. Phys., vol. 231, no. 19, page 6323 6348, 2012.

[Bischof 1998] C.H Bischof. `ADIFOR 2.0`*, User's Guide (Revision D).* 192, Argonne National Library, 1998.

[Browne 2014] P.A. Browne, C.J. Budd, C. Piccolo and M. Cullen. *Fast three dimensional r-adaptive mesh redistribution.* Journal of Computational Physics, vol. 275, pages 174 – 196, 2014.

[Bruchon 2009] J. Bruchon, H. Digonnet and T. Coupez. *Using a signed distance function for the simulation of metal forming process: formulation of the contact condition and mesh adaptation. From Lagrangian approach to an Eulerian approach.* Int. J. Numer. Meth. Engng, vol. 78, no. 8, pages 980–1008, 2009.

[C. Faure 1998] Y. Papegay C. Faure. `Odyssée`*, User's Guide version 1.7.* RT-0224, INRIA, 1998.

[Cao 2002] W. Cao, W. Huang and R.D. Russell. *A Moving Mesh Method based on the Geometric Conservation Law.* SIAM J. Sci. Comput., vol. 47, pages 118–142, 2002.

[Castro 2007] C. Castro and E. Zuazua. *Systematic Continuous Adjoint Approach to Viscous Aerodynamic Design on Unstructured Grids.* AIAA Journal, vol. 45, no. 9, pages 2125–2139, 2007.

[Causin 2005] P. Causin, J.-F. Gerbeau and F. Nobile. *Added-mass effect in the design of partitioned algorithms for fluid-structure problems.* Comput. Method Appl. Mech. Eng., pages 4506–4527, 2005.

[Chacón 2011] L. Chacón, G.L. Delzanno and J.M. Finn. *Robust, multidimensional mesh-motion based on Monge–Kantorovich equidistribution.* Journal of Computational Physics, vol. 230, no. 1, pages 87–103, 2011.

[Clément 1975] P. Clément. *Approximation by finite element functions using local regularization.* Revue Française d'Automatique, Informatique et Recherche Opérationnelle, vol. R-2, pages 77–84, 1975.

[Compère 2010] G. Compère, J.-F. Remacle, J. Jansson and J. Hoffman. *A mesh adaptation framework for dealing with large deforming meshes.* Int. J. Numer. Meth. Engng, vol. 82, pages 843–867, 2010.

[Compère 2008] G. Compère, E. Marchandise and J.-F. Remacle. *Transient adaptivity applied to two-phase incompressible flows.* Journal of Computational Physics, vol. 227, no. 3, pages 1923 – 1942, 2008.

[Constantinescu 2007] E. Constantinescu and A. Sandu. *Multirate Timestepping Methods for Hyperbolic Conservation Law.* Journal of Scientific Computing, vol. 33, pages 239–278, 2007.

[Coupez 2013] T. Coupez, G. Jannoun, N. Nassif, H.C. Nguyen, H. Digonnet and E. Hachem. *Adaptive time-step with anisotropic meshing for incompressible flows.* Journal of Computational Physics, vol. 241, no. 0, pages 195 – 211, 2013.

[Cournède 2006] P.-H. Cournède, B. Koobus and A. Dervieux. *Positivity statements for a Mixed-Element-Volume scheme on fixed and moving grids.* European Journal of Computational Mechanics, vol. 15, no. 7-8, pages 767–798, 2006.

[de Boer 2007] A. de Boer, M. van der Schoot and H. Bijl. *Mesh deformation based on radial basis function interpolation.* Comput. & Struct., vol. 85, pages 784–795, 2007.

[de Sampaio 1993] P.A. de Sampaio, P.R. Lyra, K. Morgan and N. Weatherill. *Petrov-Galerkin solutions of the incompressible Navier-Stokes equations in primitive variables with adaptive remeshing.* Comput. Methods Appl. Mech. Engrg., vol. 106, pages 143–178, 1993.

[Debiez 2000] C. Debiez and A. Dervieux. *Mixed-Element-Volume MUSCL methods with weak viscosity for steady and unsteady flow calculations.* Comput. & Fluids, vol. 29, pages 89–118, 2000.

[Degand 2002] Christoph Degand and Charbel Farhat. *A three-dimensional torsional spring analogy method for unstructured dynamic meshes.* Comput. & Struct., vol. 80, no. 3–4, pages 305 – 316, 2002.

[Denner 2014] A. Denner. *Experimental on Temporal Variable Step BDF2 Algorithms.* PhD thesis, University of Winconsin-Milwaukee, 2014.

[Donea 1982] J. Donea, S. Giuliani and J. P. Halleux. *An arbitrary Lagrangian-Eulerian finite element method for transient dynamic fluid-structure interactions.* Comput. Methods Appl. Mech. Engrg., vol. 33, no. 1, pages 689–723, 1982.

[Dvinsky 1991] A.S. Dvinsky. *Adaptive Grid Generation from Harmonic Maps on Riemannian Manifolds.* J. Comp. Phys., vol. 95, pages 450–476, 1991.

[E. Gauci 2015] A. Loseille A. Dervieux E. Gauci F. Alauzet. *Towards goal-oriented mesh adaptation for fluid-structure interaction.* In Coupled Problems in Science and Engineering, 2015.

[E. Gauci 2017] A. Dervieux E. Gauci F. Alauzet. *Vortical Flow Prediction of a Moving Aircraft Using Goal-Oriented Anisotropic Mesh Adaptation.* In 23^{rd} AIAA Computational Fluid Dynamics Conference, 2017.

[Etienne 2009] S. Etienne, A. Garon and D. Pelletier. *Perspective on the Geometric Conservation Law and Finite Element Methods for ALE Simulations of Incompressible Flow.* J. Comp. Phys., vol. 228, no. 7, pages 2313–2333, 2009.

[Etienne 2010] S. Etienne, A. Garon, D. Pelletier and C. Cameron. *Philiadium gregarium Versus Aurelia aurita: On Propulsion Propulsion of Jellyfish.* In 48th AIAA Aerospace Sciences Meeting, AIAA Paper 2010-1444, Orlando, FL, USA, Jan 2010.

[Farhat 2001] C. Farhat, P. Geuzaine and C. Grandmont. *The Discrete Geometric Conservation Law and the Nonlinear Stability of ALE Schemes for the Solution of Flow Problems on Moving Grids.* J. Comp. Phys., vol. 174, no. 2, pages 669–694, 2001.

[Fezoui 1989] L. Fezoui and A. Dervieux. *Finite-element non oscillatory schemes for compressible flows.* In Symposium on Computational Mathematics and Applications, volume 730, Pavia, Italy, 1989.

[Fidkowski 2017] K. Fidkowski. *Output-based space-time mesh optimization for unsteady flows using continuous-in-time adjoints.* J. Comp. Phys., vol. 341, pages 258–277, 2017.

[Formaggia 2004] L. Formaggia and F. Nobile. *Stability analysis of second-order time accurate schemes for ALE–FEM.* Computer Methods in Applied Mechanics and Engineering, vol. 193, no. 39–41, pages 4097 – 4116, 2004.

[Formaggia 2009] L. Formaggia, A. Quarteroni and A. Veneziani. Cardiovascular Mathematics: Modeling and simulation of the circulatory system, volume 1. Springer - Modelling, Simulations and Applications, 2009.

[Frazza 2018] L. Frazza. *3D anisotropic mesh adaptation for Reynolds Averaged Navier-Stokes simulations.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2018.

[Frey 2008] P.J. Frey and P.L. George. Mesh generation. Application to finite elements. ISTE Ltd and John Wiley & Sons, 2nd édition, 2008.

[Froehle 2014] B. Froehle and P.-O. Persson. *High-order accurate fluid–structure simulation of a tuning fork.* Computers & Fluids, vol. 98, pages 230–238, 2014.

[F.X. Le Dimet 1986] O. Talagrand F.X. Le Dimet. *Variational algorithm for analysis and assimilation of meteorological observations : theoretical aspects.* Tellus, vol. 38A, pages 97–110, 1986.

[George 1991] P.L. George, F. Hecht and M.G. Vallet. *Creation of internal points in Voronoi's type method. Control and adaptation.* Adv. Eng. Software, vol. 13, no. 5-6, pages 303–312, 1991.

[Gerbeau 2014] J.F. Gerbeau, D. Lombardi and E. Schenone. *Reduced order model in cardiac electrophysiology with approximated Lax pairs.* Advances in Computational Mathematics, pages 1–28, 2014.

[Giles 1999] M.B. Giles and N.A. Pierce. *Improved lift and drag estimates using adjoint Euler equations.* AIAA paper, vol. 99-3293, 1999.

[Giles 2000] M.B. Giles and N.A. Pierce. *An introduction to the adjoint approach to design.* Flow, Turbulence and Combustion, vol. 65, pages 393–415, 2000.

[Giles 2002] M.B. Giles and E. Suli. Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality, pages 145–236. Cambridge University Press, 2002.

[Guardone 2011] A. Guardone, D. Isola and G. Quaranta. *Arbitrary Lagrangian Eulerian formulation for two-dimensional flows using dynamic meshes with edge swapping.* Journal of Computational Physics, vol. 230, no. 20, pages 7706 – 7722, 2011.

[Harten 1983] A. Harten, P.D. Lax and B. Van Leer. *On upstream differencing and godunov-type schemes for hyperbolic conservation laws.* SIAM Revue, vol. 25, no. 1, pages 35–61, 1983.

[Hascoet 2004] L. Hascoet and V. Pascual. *Tapenade 2.1 user's guide.* RT-0300, INRIA, September 2004.

[Hassan 2007] O. Hassan, K.A. Sørensen, K. Morgan and N. P. Weatherill. *A method for time accurate turbulent compressible fluid flow simulation with moving boundary components employing local remeshing.* Int. J. Numer. Meth. Fluids, vol. 53, no. 8, pages 1243–1266, 2007.

[Hay 2014] A. Hay, K.R. Yu, S. Etienne, A. Garon and D. Pelletier. *High-order temporal accuracy for 3D finite-element ALE flow simulations.* Comput. & Fluids, vol. 100, no. 0, pages 204–217, 2014.

[Hirt 1974] C. W. Hirt, A. A. Amsden and J. L. Cook. *An arbitrary Lagrangian-Eulerian computing method for all flow speeds.* J. Comp. Phys., vol. 14, no. 3, pages 227–253, 1974.

[Huang 2010a] W. Huang, L. Kamenski and X. Li. *A new anisotropic mesh adaptation method based upon hierarchical a posteriori error estimates.* J. Comp. Phys., vol. 229, pages 2179–2198, 2010.

[Huang 2010b] W. Huang and R. D. Russell. Adaptive moving mesh methods, volume 174. Springer Science & Business Media, 2010.

[Hughes 1981] T. J. R. Hughes, W. K. Liu and T. K. Zimmermann. *Lagrangian-Eulerian finite element formulation for incompressible viscous flows.* Comput. Methods Appl. Mech. Engrg., vol. 29, no. 3, pages 329–349, 1981.

[Itam 2017] E. Itam. *Simulation numérique d'écoulements autour de corps non profilés par des modèles de turbulence hybridest un schéma multirate.* PhD thesis, Université de Montpellier, Montpellier, France, 2017.

[J. Berland 2007] O. Marsden C. Bailly J. Berland C. Bogey. *High-order, low dispersive and low dissipative explicit schemes for multi-scale and boundary problems.* J. Comp. Phys., vol. 224, pages 637–662, 2007.

[J. F. Thompson 1999] N. P. Weatherill J. F. Thompson B. K. Soni. Handbook of grid generation. CRC Press, 1999.

[J. Utke 2008] M. Fagan N. Tallent M. Strout P. Heimbach C. Hill C. Wunsch J. Utke U. Naumann. *OpenAD/F : A Modular Open-Source Tool for Automatic Differentiation of Fortran Codes.* ACM Transactions on Mathematical Software, vol. 34, 2008.

[Johnen 2013] A. Johnen, J-F. Remacle and C. Geuzaine. *Geometrical validity of curvilinear finite elements.* J. Comp. Phys., vol. 233, pages 359–372, 2013.

[Koobus 1999] B. Koobus and C. Farhat. *Second-order time-accurate and geometrically conservative implicit schemes for flow computations on unstructured dynamic meshes.* Comput. Methods Appl. Mech. Engrg., vol. 170, no. 1-2, pages 103–129, 1999.

[Koren 1993] B. Koren. *A robust upwind discretization method for advection, diffusion and source termes.* Notes Numer. Fluid Mech., vol. 45, pages 117–138, 1993.

[Kucharik 2008] M. Kucharik and M. Shashkov. *Extension of Efficient, Swept-Integration-Based Conservative Method for Meshes with Changing Connectivity.* Int. J. Numer. Meth. Fluids, vol. 56, no. 8, pages 1359–1365, 2008.

[Leer 1972] B. Van Leer. *Towards the ultimate conservative difference scheme I. The quest of monotonicity.* Lecture notes in physics, vol. 18, page 163, 1972.

[Lesoinne 1996] M. Lesoinne and C. Farhat. *Geometric Conservation Laws for Flow Problems with Moving Boundaries and Deformable Meshes, and their Impact on Aeroelastic Computations.* Comput. Methods Appl. Mech. Engrg., vol. 134, no. 1-2, pages 71–90, 1996.

[Lions 1971] J.L. Lions. Optimal control of systems governed by partial differential equations. Springer, Berlin, 170 édition, 1971.

[Löhner 1990] R. Löhner. *Three-dimensional fluid-structure interaction using a finite element solver and adaptive remeshing.* Computing Systems in Engineering, vol. 1, no. 2-4, pages 257–272, 1990.

[Löhner 1992] R. Löhner and J.D. Baum. *Adaptive h-refinement on 3D unstructured grids for transient problems.* Int. J. Numer. Meth. Fluids, vol. 14, no. 12, pages 1407–1419, 1992.

[Löhner 1998] R. Löhner and C. Yang. *Improved ALE mesh velocities for moving bodies.* Communications in numerical methods in engineering, no. 12, pages 599–608, 1998.

[Löhner 2001] R. Löhner. Applied CFD techniques. An introduction based on finite element methods. John Wiley & Sons, Ltd, New York, 2001.

[Loseille 2008] A. Loseille. *Adaptation de maillage anisotrope 3D multi-échelles et ciblée à une fonctionnelle pour la mécanique des fluides. Application à la prédiction haute-fidélité du bang sonique.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2008. (in French).

[Loseille 2010a] A. Loseille and F. Alauzet. *Optimal 3D highly anisotropic mesh adaptation applied to steady flows computation.* J. Comp. Phys., 2010. Submitted.

[Loseille 2010b] A. Loseille, A. Dervieux and F. Alauzet. *Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations.* Journal of Computational Physics, vol. 229, pages 2866–2897, 2010.

[Loseille 2010c] A. Loseille, A. Dervieux and F. Alauzet. *Fully anisotropic goal-oriented mesh adaptation for 3D steady Euler equations.* J. Comp. Phys., vol. 229, pages 2866–2897, 2010.

[Loseille 2011a] A. Loseille and F. Alauzet. *Continuous mesh framework. Part I: well-posed continuous interpolation error.* SIAM J. Numer. Anal., vol. 49, no. 1, pages 38–60, 2011.

[Loseille 2011b] A. Loseille and F. Alauzet. *Continuous mesh framework. Part II: validations and applications.* SIAM J. Numer. Anal., vol. 49, no. 1, pages 61–86, 2011.

[Loseille 2013] A. Loseille and V. Menier. *Serial and Parallel Mesh Modification Through a Unique Cavity-Based Primitive.* In Proceedings of the 22th International Meshing Roundtable, pages 541–558. Springer, Orlando, Fl, USA, 2013.

[Loseille 2017] A. Loseille, F. Alauzet and V. Menier. *Unique cavity-based operator and hierarchical domain partitioning for fast parallel generation of anisotropic meshes.* Computer-Aided Design, vol. 85, pages 53–67, 2017.

[Loubère 2010] R. Loubère, P-H. Maire, M. Shashkov, J. Breil and S. Galera. *ReALE: a Reconnaction-Based Arbitrary-Lagrangian-Eulerian Method.* J. Comp. Phys., vol. 229, pages 4724–4761, 2010.

[Luke 2012] E. Luke, E. Collins and E. Blades. *A fast mesh deformation method using explicit interpolation.* J. Comp. Phys., vol. 231, pages 586–601, 2012.

[M. Martinelli 2010] L. Hascoët V. Pascual M. Martinelli A. Dervieux and A. Belme. *AD-based perturbation methods for uncertainties and errors.* Int. J. Engineering Systems Modelling and simulation, vol. 2, pages 65–74, 2010.

[Marcum 2001] D. L. Marcum. *Efficient Generation of High-Quality Unstructured Surface and Volume Grids.* Engineering with Computers, vol. 17, no. 3, pages 211–233, Oct 2001.

[Maréchal 2016] L. Maréchal. *Handling unstructured meshes in multithreaded environments with the help of Hilbert renumbering and dynamic scheduling.* Parallel Computing, to be published 2016.

[Martin 1996] R. Martin and H. Guillard. *A second order defect correction scheme for unsteady problems.* Comput. & Fluids, vol. 25, pages 9–27, 1996.

[Mavriplis 2000] D.J. Mavriplis. *Adaptive Meshing Techniques for Viscous Flow Calculations on Mixed Element Unstructured Meshes.* Int. J. Numer. Meth. Fluids, vol. 34, no. 2, pages 93–111, 2000.

[Mavriplis 2006] D.J. Mavriplis and Z. Yang. *Construction of the Discrete Geometric Conservation Law for High-Order Time Accurate Simulations on Dynamic Meshes.* J. Comp. Phys., vol. 213, no. 2, pages 557–573, 2006.

[Menier 2015] V. Menier. *Mesh Adaptation For the High Fidelity Prediction of Viscous Phenomena and Their Interactions. Application to Aeronautics.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2015.

[Miller 1981] K. Miller and R. N. Miller. *Moving Finite Elements. I.* SIAM Journal on Numerical Analysis, vol. 18, no. 6, pages 1019–1032, 1981.

[Mohammadi 1997] B. Mohammadi. *Optimal shape design, reverse mode of automatic differentiation and turbulence.* AIAA, 1997.

[Murman 2003] S. Murman, M. Aftosmis and M. Berger. *Simulation of 6-DOF Motion with Cartesian Method.* In 41th AIAA Aerospace Sciences Meeting and Exhibit, AIAA-2003-1246, Reno, NV, USA, Jan 2003.

[N. Barral 2017] F. Alauzet N. Barral G. Olivier. *Time-accurate anisotropic mesh adaptation for three-dimensional time-dependent problems with body-fitted moving geometries.* J. Comp. Phys., vol. 331, pages 157 – 187, 2017.

[N. Gourdain 2009] L. Gicquel et al. N. Gourdain. *High Performance Parallel Computing of Flows in Complex Geometries.* J. of Computational Science and Discovery, 2009.

[N.A. Pierce 2004] M.B. Giles N.A. Pierce. *Adjoint and defect error bounding and correction for functional estimates.* J. Comp. Phys., vol. 200, pages 769–794, 2004.

[Nkonga 1993] B. Nkonga and H. Guillard. *Godunov Type Methods on Non-Structured Meshes for Three-Dimensional Moving Boundary Problems.* RR-1883, INRIA, 1993.

[Nkonga 1994] B. Nkonga and H. Guillard. *Godunov type method on non-structured meshes for three-dimensional moving boundary problems.* Computer Methods in Applied Mechanics and Engineering, vol. 113, no. 1–2, pages 183 – 204, 1994.

[Nkonga 2000] B. Nkonga. *On the Conservative and Accurate CFD Approximations for Moving Meshes and Moving Boundaries.* Comput. Methods Appl. Mech. Engrg., vol. 190, no. 13, pages 1801–1825, 2000.

[Olivier 2011a] G. Olivier. *Unsteady metric-based mesh adaptation applied to CFD moving domain simulations.* PhD thesis, Université Pierre et Marie Curie, Paris VI, Paris, France, 2011.

[Olivier 2011b] G. Olivier and F. Alauzet. *A New Changing-Topology ALE Scheme for Moving Mesh Unsteady Simulations.* Jan 2011.

[Pain 2001] C.C Pain, A.P. Humpleby, C.R.E. de Oliveira and A.J.H. Goddard. *Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations.* Comput. Methods Appl. Mech. Engrg., vol. 190, pages 3771–3796, 2001.

[Pendenza 2014] A. Pendenza, W. G. Habashi and M. Fossati. *A 3D mesh deformation technique for irregular in-flight ice accretion.* In 44th AIAA Fluid Dynamics Conference, AIAA Paper 2014-3072, Atlanta, GA, USA, June 2014.

[Picasso 2003] M. Picasso. *An anisotropic error indicator based on Zienkiewicz-Zhu error estimator: Application to elliptic and parabolic problems.* SIAM J. Sci. Comput., vol. 24, no. 4, pages 1328–1355, 2003.

[Picasso 2009] M. Picasso, V. Prachittham and M.A.M. Gijs. *Adaptive Finite Elements with Large Aspect Ratio for Mass Transport in Electro-osmosis and Pressure-Driven Microflows.* Int. J. Numer. Meth. Fluids, 2009.

[R. Becker 2001] R. Rannacher R. Becker. *An optimal control approach to a posteriori error estimation in finite element methods.* Acta Numerica, vol. 19, pages 1–102, 2001.

[Rausch 1992] R.D. Rausch, J.T. Batina and H.T.Y. Yang. *Spatial adaptation procedures on tetrahedral meshes for unsteady aerodynamic flow calculations.* AIAA Journal, vol. 30, pages 1243–1251, 1992.

[Remacle 2005] J.-F. Remacle, X. Li, M.S. Shephard and J.E. Flaherty. *Anisotropic adaptive simulation of transient flows using discontinuous Galerkin methods.* Int. J. Numer. Meth. Engng, vol. 62, pages 899–923, 2005.

[Roe 1981] P. Roe. *Approximate Riemann solvers, parameter vectors, and difference schemes.* J. Comp. Phys., vol. 43, pages 357–372, 1981.

[Rusanov 1961] V. Rusanov. *Calculation of Intersection of Non-Steady Shock Waves with Obstacles.* J. Comp. Phys., pages 267–279, 1961.

[S. Piperno 1998] S. Depeyre S. Piperno. *Criteria for the design of limiters yielding efficient high resolution TVD schemes.* Computers & Fluids, vol. 27, pages 183–197, 1998.

[Sagan 1994] H. Sagan. Space-filling curves. Springer, New York, NY, 1994.

[Saksono 2007] P.H. Saksono, W.G. Dettmer and D. Perić. *An Adaptive Remeshing Strategy for Flows with Moving Boundaries and Fluid-Structure Interaction.* Int. J. Numer. Meth. Engng, vol. 71, no. 9, pages 1009–1050, 2007.

[Shu 1988] C.W. Shu and S. Osher. *Efficient implementation of essentially non-oscillatory shock-capturing schemes.* J. Comput. Phys., vol. 77, pages 439–471, 1988.

[Speares 1997] W. Speares and M. Berzins. *A 3D unstructured mesh adaptation algorithm for time-dependent shock-dominated problems.* Int. J. Numer. Meth. Fluids, vol. 25, pages 81–104, 1997.

[Spiteri 2002] R.J. Spiteri and S.J. Ruuth. *A new class of optimal high-order strong-stability-preserving time discretization methods.* SIAM J. Numer. Anal., vol. 40, no. 2, pages 469–491, 2002.

[Stoufflet 1987] B. Stoufflet, J. Periaux, L. Fezoui and A. Dervieux. *Numerical simulation of 3-D hypersonic Euler flows around space vehicles using adapted finite element.* In AIAA 25th Aerospace Sciences Meeting, AIAA-1987-0560, Reno, NV, USA, Jan 1987.

[Thompson 1983] J.F Thompson and Z.U.A. Warsi. *Three-Dimensional Mesh Generation from Elliptic Systems.* In Proceedings of the AIAA CFD Conference, volume AIAA-83-1905, Danvers, USA, July 1983.

[Tong 2014] X. Tong, D. Thompson, Q. Arnoldus, E. Collins and E. Luke. *Robust surface evolution and mesh deformation for three dimensional aircraft icing applications on a swept GLC-305 airfoil.* In 6th AIAA Atmospheric and Space Environments Conference, AIAA Paper 2014-2201, Atlanta, GA, USA, June 2014.

[Venditti 2002] D.A. Venditti and D.L. Darmofal. *Grid adaptation for functional outputs: application to two-dimensional inviscid flows.* J. Comp. Phys., vol. 176, no. 1, pages 40–69, 2002.

[Walter] S. Walter. *PyADOL-C : a python module to differentiate complex algorithms written in Python.* http://www.github.com/b45ch1/pyadolc/.

[Winslow 1963] A. Winslow. *Equipotential Zoning of Two Dimensional Meshes.* Lawrence Livemore Laboratory, California, USA, 1963. Technical Report UCRL-7312.

[Wintzer 2008] M. Wintzer, M. Nemec and M.J. Aftosmis. *Adjoint-Based Adaptive Mesh Refinement for Sonic Boom Prediction.* In AIAA 26th Applied Aerodynamics Conference, AIAA-2008-6593, Honolulu, HI, USA, Aug 2008.

[Wu 1990] J. Wu, J.Z. Zhu, J. Szmelter and O.C. Zienkiewicz. *Error estimation and adaptivity in Navier-Stokes incompressible flows.* Computational Mechanics, vol. 6, pages 259–270, 1990.

[Yang 2005] Z. Yang and D.J. Mavriplis. *Unstructured Dynamic Meshes with Higher-Order Time Integration Schemes for the Unsteady Navier-Stokes Equations.* In 41th AIAA Aerospace Sciences Meeting, AIAA Paper 2005-1222, Reno, NV, USA, Jan 2005.

[Yang 2007] Z. Yang and D.J. Mavriplis. *Higher-Order Time Integration Schemes for Aeroelastic Applications on Unstructured Meshes.* AIAA Journal, vol. 45, no. 1, pages 138–150, 2007.

[Zienkiewicz 1992] O.C. Zienkiewicz and J.Z. Zhu. *The superconvergent patch recovery and a posteriori error estimates. Part 1: The recovery technique.* Int. J. Numer. Meth. Engng, vol. 33, no. 7, pages 1331–1364, 1992.